JOURNAL OF FORMALIZED MATHEMATICS
Volume 10, Released 1998, Published 2003
Inst. of Computer Science, Univ. of Białystok

Initialization Halting Concepts and Their Basic Properties of $\mathbf{SCM}_{\mathrm{FSA}}$

Jing-Chao Chen
Shanghai Jiaotong University

Yatsuka Nakamura
Shinshu University
Nagano

Summary. Up to now, many properties of macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ are described by the parahalting concepts. However, many practical programs are not always halting while they are halting for initialization states. For this reason, we propose initialization halting concepts. That a program is initialization halting (called "InitHalting" for short) means it is halting for initialization states. In order to make the halting proof of more complicated programs easy, we present "InitHalting" basic properties of the compositions of the macro instructions, if-Macro (conditional branch macro instructions) and Times-Macro (for-loop macro instructions) etc.

MML Identifier: `SCM_HALT`.

WWW: `http://mizar.org/JFM/Vol10/scm_halt.html`

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

For simplicity, we use the following convention: m is a natural number, I is a macro instruction, s, s_1, s_2 are states of $\mathbf{SCM}_{\mathrm{FSA}}$, a is an integer location, and f is a finite sequence location.

Let I be a macro instruction. We say that I is InitClosed if and only if:

(Def. 1) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every natural number n such that $\mathrm{Initialized}(I) \subseteq s$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(n)} \in \mathrm{dom} I$.

We say that I is InitHalting if and only if:

(Def. 2) $\mathrm{Initialized}(I)$ is halting.

We say that I is keepInt0 1 if and only if:

(Def. 3) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $\mathrm{Initialized}(I) \subseteq s$ and for every natural number k holds $(\mathrm{Computation}(s))(k)(\mathrm{intloc}(0)) = 1$.

We now state two propositions:

(1) For every set x and for all natural numbers i, m, n such that $x \in \mathrm{dom}((\mathrm{intloc}(i) \mapsto m) + \cdot\, \mathrm{Start\text{-}At}(\mathrm{insloc}(n)))$ holds $x = \mathrm{intloc}(i)$ or $x = \mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(2) For every macro instruction I and for all natural numbers i, m, n holds $\mathrm{dom} I$ misses $\mathrm{dom}((\mathrm{intloc}(i) \mapsto m) + \cdot\, \mathrm{Start\text{-}At}(\mathrm{insloc}(n)))$.

We now state two propositions:

(3) Initialized$(I) = I +\cdot ((\mathrm{intloc}(0) \mapsto 1) +\cdot \text{Start-At}(\mathrm{insloc}(0)))$.

(4) Macro($\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$) is InitHalting.

Let us note that there exists a macro instruction which is InitHalting.
Next we state three propositions:

(5) For every InitHalting macro instruction I such that Initialized$(I) \subseteq s$ holds s is halting.

(6) $I +\cdot$ Start-At(insloc(0)) $\subseteq$ Initialized(I).

(7) For every macro instruction I and for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ such that Initialized$(I) \subseteq s$ holds $s(\mathrm{intloc}(0)) = 1$.

Let us observe that every macro instruction which is paraclosed is also InitClosed.
Let us observe that every macro instruction which is parahalting is also InitHalting.
One can verify the following observations:

* every macro instruction which is InitHalting is also InitClosed,
* every macro instruction which is keepInt0 1 is also InitClosed, and
* every macro instruction which is keeping 0 is also keepInt0 1.

The following two propositions are true:

(8) Let I be an InitHalting macro instruction and a be a read-write integer location. If $a \notin$ UsedIntLoc(I), then $(\mathrm{IExec}(I,s))(a) = s(a)$.

(9) Let I be an InitHalting macro instruction and f be a finite sequence location. If $f \notin$ UsedInt* Loc(I), then $(\mathrm{IExec}(I,s))(f) = s(f)$.

Let I be an InitHalting macro instruction. Observe that Initialized(I) is halting.
Let us note that every macro instruction which is InitHalting is also non empty.
One can prove the following propositions:

(10) For every InitHalting macro instruction I holds $\mathrm{dom}\, I \neq \emptyset$.

(11) For every InitHalting macro instruction I holds $\mathrm{insloc}(0) \in \mathrm{dom}\, I$.

(12) Let J be an InitHalting macro instruction. Suppose Initialized$(J) \subseteq s_1$. Let n be a natural number. Suppose ProgramPart(Relocated(J,n)) $\subseteq s_2$ and $\mathbf{IC}_{(s_2)} = \mathrm{insloc}(n)$ and $s_1 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$. Let i be a natural number. Then $\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)} + n = \mathbf{IC}_{(\mathrm{Computation}(s_2))(i)}$ and IncAddr(CurInstr((Computation(s_1))(i))$,n$) $=$ CurInstr((Computation(s_2))(i)) and (Computation(s_1))$(i) \restriction$(Int-Locations$\cup$ (Computation(s_2))$(i) \restriction$(Int-Locations$\cup$FinSeq-Locations).

(13) If Initialized$(I) \subseteq s$, then $I \subseteq s$.

(14) Let I be an InitHalting macro instruction. Suppose Initialized$(I) \subseteq s_1$ and Initialized$(I) \subseteq s_2$ and s_1 and s_2 are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Let k be a natural number. Then (Computation(s_1))(k) and (Computation(s_2))(k) are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$ and CurInstr((Computation(s_1))(k)) $=$ CurInstr((Computation(s_2))(k)).

(15) Let I be an InitHalting macro instruction. Suppose Initialized$(I) \subseteq s_1$ and Initialized$(I) \subseteq s_2$ and s_1 and s_2 are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Then LifeSpan$(s_1) =$ LifeSpan(s_2) and Result(s_1) and Result(s_2) are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(16) Macro($\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$) is keeping 0.

Let us mention that there exists a macro instruction which is keeping 0 and InitHalting.

One can verify that there exists a macro instruction which is keepInt0 1 and InitHalting.

One can prove the following propositions:

(17) For every keepInt0 1 InitHalting macro instruction I holds $(\mathrm{IExec}(I,s))(\mathrm{intloc}(0)) = 1$.

(18) Let I be an InitClosed macro instruction and J be a macro instruction. Suppose $\mathrm{Initialized}(I) \subseteq s$ and s is halting. Let given m. Suppose $m \leq \mathrm{LifeSpan}(s)$. Then $(\mathrm{Computation}(s))(m)$ and $(\mathrm{Computation}(s{+\cdot}(I;J)))(m)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(19) For all natural numbers i, m, n holds $s{+\cdot}I{+\cdot}((\mathrm{intloc}(i) \dot{\longmapsto} m){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(n))) = (s{+\cdot}((\mathrm{intloc}(i) \dot{\longmapsto} m){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(n)))){+\cdot}I$.

(20) If $(\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)) \subseteq s$, then $\mathrm{Initialized}(I) \subseteq s{+\cdot}(I{+\cdot}((\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0))))$ and $s{+\cdot}(I{+\cdot}((\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))) = s{+\cdot}I$ and $s{+\cdot}(I{+\cdot}((\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))){+\cdot}\,\mathrm{Dire}$ $s{+\cdot}\,\mathrm{Directed}(I)$.

(21) For every InitClosed macro instruction I such that $s{+\cdot}I$ is halting and $\mathrm{Directed}(I) \subseteq s$ and $(\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)) \subseteq s$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s+\cdot I)+1)} = \mathrm{insloc}(\mathrm{card}\,I)$.

(22) Let I be an InitClosed macro instruction. Suppose $s{+\cdot}I$ is halting and $\mathrm{Directed}(I) \subseteq s$ and $(\mathrm{intloc}(0) \dot{\longmapsto} 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)) \subseteq s$. Then $(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s{+\cdot}I)){\restriction}(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) =$ $(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s{+\cdot}I)+1){\restriction}(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(23) Let I be an InitHalting macro instruction. Suppose $\mathrm{Initialized}(I) \subseteq s$. Let k be a natural number. If $k \leq \mathrm{LifeSpan}(s)$, then $\mathrm{CurInstr}((\mathrm{Computation}(s{+\cdot}\,\mathrm{Directed}(I)))(k)) \neq \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(24) Let I be an InitClosed macro instruction. Suppose $s{+\cdot}\,\mathrm{Initialized}(I)$ is halting. Let J be a macro instruction and k be a natural number. Suppose $k \leq \mathrm{LifeSpan}(s{+\cdot}\,\mathrm{Initialized}(I))$. Then $(\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(I)))(k)$ and $(\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(I;J)))(k)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(25) Let I be a keepInt0 1 InitHalting macro instruction, J be an InitHalting macro instruction, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $\mathrm{Initialized}(I;J) \subseteq s$. Then $\mathbf{IC}_{(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s+\cdot I)+1)} = \mathrm{insloc}(\mathrm{card}\,I)$ and $(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s{+\cdot}I) + 1){\restriction}(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = ((\mathrm{Computation}(s{+\cdot}I))(\mathrm{LifeSpan}(s{+\cdot}I)){+\cdot}\,\mathrm{Initialized}(J)){\restriction}(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSe}$ and $\mathrm{ProgramPart}(\mathrm{Relocated}(J,\mathrm{card}\,I)) \subseteq (\mathrm{Computation}(s))(\mathrm{LifeSpan}(s{+\cdot}I)+1)$ and $(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s{+\cdot}I) + 1)(\mathrm{intloc}(0)) = 1$ and s is halting and $\mathrm{LifeSpan}(s) = \mathrm{LifeSpan}(s{+\cdot}I)+1+\mathrm{LifeSpan}(\mathrm{Result}(s{+\cdot}I){+\cdot}\,\mathrm{Initialized}(J))$ and if J is keeping 0, then $(\mathrm{Result}(s))(\mathrm{intloc}(0)) = 1$.

Let I be a keepInt0 1 InitHalting macro instruction and let J be an InitHalting macro instruction. Observe that $I;J$ is InitHalting.

Next we state four propositions:

(26) Let I be a keepInt0 1 macro instruction. Suppose $s{+\cdot}I$ is halting. Let J be an InitClosed macro instruction. Suppose $\mathrm{Initialized}(I;J) \subseteq s$. Let k be a natural number. Then $(\mathrm{Computation}(\mathrm{Result}(s{+\cdot}I){+\cdot}\,\mathrm{Initialized}(J)))(k){+\cdot}\,\mathrm{Start\text{-}At}(\mathbf{IC}_{(\mathrm{Computation}(\mathrm{Result}(s+\cdot I)+\cdot\,\mathrm{Initialized}(J)))(k)} + \mathrm{card}\,I)$ and $(\mathrm{Computation}(s{+\cdot}(I;J)))(\mathrm{LifeSpan}(s{+\cdot}I)+1+k)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(27) Let I be a keepInt0 1 macro instruction. Suppose $s{+\cdot}\,\mathrm{Initialized}(I)$ is not halting. Let J be a macro instruction and k be a natural number. Then $(\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(I)))(k)$ and $(\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(I;J)))(k)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(28) Let I be a keepInt0 1 InitHalting macro instruction and J be an InitHalting macro instruction. Then $\mathrm{LifeSpan}(s{+\cdot}\,\mathrm{Initialized}(I;J)) = \mathrm{LifeSpan}(s{+\cdot}\,\mathrm{Initialized}(I)) + 1 + \mathrm{LifeSpan}(\mathrm{Result}(s{+\cdot}\,\mathrm{Initialized}(I)){+\cdot}\,\mathrm{Initialized}(J))$.

(29) Let I be a keepInt0 1 InitHalting macro instruction and J be an InitHalting macro instruction. Then $\mathrm{IExec}(I; J, s) = \mathrm{IExec}(J, \mathrm{IExec}(I, s)) +\cdot \text{Start-At}(\mathbf{IC}_{\mathrm{IExec}(J,\mathrm{IExec}(I,s))} + \mathrm{card}\, I)$.

Let i be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. One can verify that $\mathrm{Macro}(i)$ is InitHalting.

Let i be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let J be a parahalting macro instruction. Observe that i; J is InitHalting.

Let i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let J be an InitHalting macro instruction. Observe that i; J is InitHalting.

Let I, J be keepInt0 1 macro instructions. Note that I; J is keepInt0 1.

Let j be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let I be a keepInt0 1 InitHalting macro instruction. Observe that I; j is InitHalting and keepInt0 1.

Let i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let J be a keepInt0 1 InitHalting macro instruction. Note that i; J is InitHalting and keepInt0 1.

Let j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let I be a parahalting macro instruction. Observe that I; j is InitHalting.

Let i, j be parahalting instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Note that i; j is InitHalting.

The following propositions are true:

(30) Let I be a keepInt0 1 InitHalting macro instruction and J be an InitHalting macro instruction. Then $(\mathrm{IExec}(I; J, s))(a) = (\mathrm{IExec}(J, \mathrm{IExec}(I, s)))(a)$.

(31) Let I be a keepInt0 1 InitHalting macro instruction and J be an InitHalting macro instruction. Then $(\mathrm{IExec}(I; J, s))(f) = (\mathrm{IExec}(J, \mathrm{IExec}(I, s)))(f)$.

(32) For every keepInt0 1 InitHalting macro instruction I and for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{Initialize}(\mathrm{IExec}(I, s)) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = \mathrm{IExec}(I, s) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$.

(33) Let I be a keepInt0 1 InitHalting macro instruction and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(I; j, s))(a) = (\mathrm{Exec}(j, \mathrm{IExec}(I, s)))(a)$.

(34) Let I be a keepInt0 1 InitHalting macro instruction and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(I; j, s))(f) = (\mathrm{Exec}(j, \mathrm{IExec}(I, s)))(f)$.

Let I be a macro instruction and let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. We say that I is closed onInit s if and only if:

(Def. 4) For every natural number k holds $\mathbf{IC}_{(\mathrm{Computation}(s +\cdot \mathrm{Initialized}(I)))(k)} \in \mathrm{dom}\, I$.

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

(Def. 5) $s +\cdot \mathrm{Initialized}(I)$ is halting.

The following propositions are true:

(35) Let I be a macro instruction. Then I is InitClosed if and only if for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds I is closed onInit s.

(36) Let I be a macro instruction. Then I is InitHalting if and only if for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds I is halting onInit s.

(37) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be an integer location. Suppose I does not destroy a and I is closed onInit s and $\mathrm{Initialized}(I) \subseteq s$. Let k be a natural number. Then $(\mathrm{Computation}(s))(k)(a) = s(a)$.

Let us observe that there exists a macro instruction which is InitHalting and good.

Let us mention that every macro instruction which is InitClosed and good is also keepInt0 1.

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

One can prove the following propositions:

(38) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, J be an InitHalting macro instruction, and a be an integer location. Then $(\mathrm{IExec}(i; J, s))(a) = (\mathrm{IExec}(J, \mathrm{Exec}(i, \mathrm{Initialize}(s))))(a)$.

(39) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\rm FSA}$, J be an InitHalting macro instruction, and f be a finite sequence location. Then $({\rm IExec}(i;J,s))(f) = ({\rm IExec}(J,{\rm Exec}(i,{\rm Initialize}(s))))(f)$.

(40) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Then I is closed onInit s if and only if I is closed on Initialize(s).

(41) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Then I is halting onInit s if and only if I is halting on Initialize(s).

(42) For every macro instruction I and for every state s of $\mathbf{SCM}_{\rm FSA}$ holds ${\rm IExec}(I,s) = {\rm IExec}(I,{\rm Initialize}(s))$.

(43) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) = 0$ and I is closed onInit s and I is halting onInit s. Then **if** $a = 0$ **then** I **else** J is closed onInit s and **if** $a = 0$ **then** I **else** J is halting onInit s.

(44) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) = 0$ and I is closed onInit s and I is halting onInit s. Then IExec(**if** $a = 0$ **then** I **else** $J,s) = {\rm IExec}(I,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$.

(45) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) \neq 0$ and J is closed onInit s and J is halting onInit s. Then **if** $a = 0$ **then** I **else** J is closed onInit s and **if** $a = 0$ **then** I **else** J is halting onInit s.

(46) Let I, J be macro instructions, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\rm FSA}$. Suppose $s(a) \neq 0$ and J is closed onInit s and J is halting onInit s. Then IExec(**if** $a = 0$ **then** I **else** $J,s) = {\rm IExec}(J,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$.

(47) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be InitHalting macro instructions, and a be a read-write integer location. Then **if** $a = 0$ **then** I **else** J is InitHalting and if $s(a) = 0$, then IExec(**if** $a = 0$ **then** I **else** $J,s) = {\rm IExec}(I,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$ and if $s(a) \neq 0$, then IExec(**if** $a = 0$ **then** I **else** $J,s) = {\rm IExec}(J,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$.

(48) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be InitHalting macro instructions, and a be a read-write integer location. Then

(i) $\mathbf{IC}_{{\rm IExec}(\mathbf{if}\ a=0\ \mathbf{then}\ I\ \mathbf{else}\ J,s)} = {\rm insloc}({\rm card}\,I + {\rm card}\,J + 3)$,

(ii) if $s(a) = 0$, then for every integer location d holds (IExec(**if** $a = 0$ **then** I **else** $J,s))(d) = ({\rm IExec}(I,s))(d)$ and for every finite sequence location f holds (IExec(**if** $a = 0$ **then** I **else** $J,s))(f) = ({\rm IExec}(I,s))(f)$, and

(iii) if $s(a) \neq 0$, then for every integer location d holds (IExec(**if** $a = 0$ **then** I **else** $J,s))(d) = ({\rm IExec}(J,s))(d)$ and for every finite sequence location f holds (IExec(**if** $a = 0$ **then** I **else** $J,s))(f) = ({\rm IExec}(J,s))(f)$.

(49) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) > 0$ and I is closed onInit s and I is halting onInit s. Then **if** $a > 0$ **then** I **else** J is closed onInit s and **if** $a > 0$ **then** I **else** J is halting onInit s.

(50) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) > 0$ and I is closed onInit s and I is halting onInit s. Then IExec(**if** $a > 0$ **then** I **else** $J,s) = {\rm IExec}(I,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$.

(51) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) \leq 0$ and J is closed onInit s and J is halting onInit s. Then **if** $a > 0$ **then** I **else** J is closed onInit s and **if** $a > 0$ **then** I **else** J is halting onInit s.

(52) Let I, J be macro instructions, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\rm FSA}$. Suppose $s(a) \leq 0$ and J is closed onInit s and J is halting onInit s. Then IExec(**if** $a > 0$ **then** I **else** $J,s) = {\rm IExec}(J,s) {+\cdot}\, \text{Start-At}({\rm insloc}({\rm card}\,I + {\rm card}\,J + 3))$.

(53) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be InitHalting macro instructions, and a be a read-write integer location. Then **if** $a > 0$ **then** I **else** J is InitHalting and if $s(a) > 0$, then IExec(**if** $a > 0$ **then** I **else** J, s) $=$ IExec(I, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + 3)) and if $s(a) \leq 0$, then IExec(**if** $a > 0$ **then** I **else** J, s) $=$ IExec(J, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + 3)).

(54) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be InitHalting macro instructions, and a be a read-write integer location. Then

(i) $\mathbf{IC}_{\mathrm{IExec}(\mathbf{if}\ a>0\ \mathbf{then}\ I\ \mathbf{else}\ J,s)} = \mathrm{insloc}(\mathrm{card}\,I + \mathrm{card}\,J + 3)$,

(ii) if $s(a) > 0$, then for every integer location d holds (IExec(**if** $a > 0$ **then** I **else** J, s))$(d) =$ (IExec(I, s))(d) and for every finite sequence location f holds (IExec(**if** $a > 0$ **then** I **else** J, s))$(f) =$ (IExec(I, s))(f), and

(iii) if $s(a) \leq 0$, then for every integer location d holds (IExec(**if** $a > 0$ **then** I **else** J, s))$(d) =$ (IExec(J, s))(d) and for every finite sequence location f holds (IExec(**if** $a > 0$ **then** I **else** J, s))$(f) =$ (IExec(J, s))(f).

(55) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) < 0$ and I is closed onInit s and I is halting onInit s. Then IExec(**if** $a < 0$ **then** I **else** J, s) $=$ IExec(I, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + card$J$ + 7)).

(56) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) = 0$ and J is closed onInit s and J is halting onInit s. Then IExec(**if** $a < 0$ **then** I **else** J, s) $=$ IExec(J, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + card$J$ + 7)).

(57) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) > 0$ and J is closed onInit s and J is halting onInit s. Then IExec(**if** $a < 0$ **then** I **else** J, s) $=$ IExec(J, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + card$J$ + 7)).

(58) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be InitHalting macro instructions, and a be a read-write integer location. Then

(i) **if** $a < 0$ **then** I **else** J is InitHalting,

(ii) if $s(a) < 0$, then IExec(**if** $a < 0$ **then** I **else** J, s) $=$ IExec(I, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + card$J$ + 7)), and

(iii) if $s(a) \geq 0$, then IExec(**if** $a < 0$ **then** I **else** J, s) $=$ IExec(J, s)$+\cdot$ Start-At(insloc(card$I$ + card$J$ + card$J$ + 7)).

Let I, J be InitHalting macro instructions and let a be a read-write integer location. One can verify the following observations:

* **if** $a = 0$ **then** I **else** J is InitHalting,
* **if** $a > 0$ **then** I **else** J is InitHalting, and
* **if** $a < 0$ **then** I **else** J is InitHalting.

Next we state four propositions:

(59) For every macro instruction I holds I is InitHalting iff for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds I is halting on Initialize(s).

(60) For every macro instruction I holds I is InitClosed iff for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds I is closed on Initialize(s).

(61) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be an InitHalting macro instruction, and a be a read-write integer location. Then (IExec(I, s))$(a) =$ (Computation(Initialize(s)$+\cdot$($I+\cdot$ Start-At(insloc(0)))))(LifeSpan(Initialize(s)$+\cdot$($I+$

(62) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be an InitHalting macro instruction, a be an integer location, and k be a natural number. If I does not destroy a, then (IExec(I, s))$(a) =$ (Computation(Initialize(s)$+\cdot$($I+\cdot$ Start-At(insloc(0)))))$(k)(a)$.

The following propositions are true:

(63) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be an InitHalting macro instruction, and a be an integer location. If I does not destroy a, then $({\rm IExec}(I,s))(a) = ({\rm Initialize}(s))(a)$.

(64) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a keepInt0 1 InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a. Then $({\rm Computation}({\rm Initialize}(s)+\cdot((I;\ {\rm SubFrom}(a,{\rm intloc}(0)))+\cdot\ {\rm Start\text{-}At}({\rm insloc}(0)))))({\rm LifeSpan}({\rm Initialize}(s)+\cdot((I;\ {\rm SubFrom}($ $s(a)-1$.

(65) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be an InitClosed macro instruction. Suppose ${\rm Initialized}(I) \subseteq s$ and s is halting. Let m be a natural number. Suppose $m \leq {\rm LifeSpan}(s)$. Then $({\rm Computation}(s))(m)$ and $({\rm Computation}(s+\cdot\ {\rm loop}\,I))(m)$ are equal outside the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(66) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be an InitHalting macro instruction. Suppose ${\rm Initialized}(I) \subseteq s$. Let k be a natural number. If $k \leq {\rm LifeSpan}(s)$, then ${\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm loop}\,I))(k)) \neq \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$.

(67) $I \subseteq s+\cdot\ {\rm Initialized}(I)$.

(68) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Suppose I is closed onInit s and I is halting onInit s. Let m be a natural number. Suppose $m \leq {\rm LifeSpan}(s+\cdot\ {\rm Initialized}(I))$. Then $({\rm Computation}(s+\cdot\ {\rm Initialized}(I)))(m)$ and $({\rm Computation}(s+\cdot\ {\rm Initialized}({\rm loop}\,I)))(m)$ are equal outside the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(69) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Suppose I is closed onInit s and I is halting onInit s. Let m be a natural number. If $m < {\rm LifeSpan}(s+\cdot\ {\rm Initialized}(I))$, then ${\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm Initialized}(I)))(m)) = {\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm Initialized}({\rm loop}\,I)))(m))$.

(70) For every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds $l \notin {\rm dom}(({\rm intloc}(0) \dot{\longmapsto} 1)+\cdot\ {\rm Start\text{-}At}({\rm insloc}(0)))$.

(71) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Suppose I is closed onInit s and I is halting onInit s. Then ${\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm Initialized}({\rm loop}\,I)))({\rm LifeSpan}(s+\cdot\ {\rm Initialized}(I)))) =$ goto insloc(0) and for every natural number m such that $m \leq {\rm LifeSpan}(s+\cdot\ {\rm Initialized}(I))$ holds ${\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm Initialized}({\rm loop}\,I)))(m)) \neq \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$.

(72) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Suppose I is closed onInit s and I is halting onInit s. Then ${\rm CurInstr}(({\rm Computation}(s+\cdot\ {\rm Initialized}({\rm loop}\,I)))({\rm LifeSpan}(s+\cdot\ {\rm Initialized}(I)))) =$ goto insloc(0).

(73) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s({\rm intloc}(0)) = 1$ and $s(a) > 0$. Then loop **if** $a = 0$ **then** Goto(insloc(2)) **else** $(I;\ {\rm SubFrom}(a,{\rm intloc}(0)))$ is pseudo-closed on s.

(74) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s(a) > 0$. Then Initialized(loop **if** $a = 0$ **then** Goto(insloc(2)) **else** $(I;\ {\rm SubFrom}(a,{\rm intloc}(0)))$) is pseudo-closed on s.

(75) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s({\rm intloc}(0)) = 1$. Then ${\rm Times}(a,I)$ is closed on s and ${\rm Times}(a,I)$ is halting on s.

(76) Let I be a good InitHalting macro instruction and a be a read-write integer location. If I does not destroy a, then ${\rm Initialized}({\rm Times}(a,I))$ is halting.

Let a be a read-write integer location and let I be a good macro instruction. Observe that ${\rm Times}(a,I)$ is good.

The following propositions are true:

(77) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s(a) > 0$. Then there exists a state s_2 of $\mathbf{SCM}_{\rm FSA}$ and there exists a natural number k such that

$s_2 = s +\cdot\, \mathrm{Initialized}(\mathrm{loop}\,\mathbf{if}\ a = 0\ \mathbf{then}\ \mathrm{Goto}(\mathrm{insloc}(2))\ \mathbf{else}\ (I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0))))$ and $k = \mathrm{LifeSpan}(s +\cdot\, \mathrm{Initialized}(\mathbf{if}\ a = 0\ \mathbf{then}\ \mathrm{Goto}(\mathrm{insloc}(2))\ \mathbf{else}\ (I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0))))) + 1$ and $(\mathrm{Computation}(s_2))(k)(a) = s(a) - 1$ and $(\mathrm{Computation}(s_2))(k)(\mathrm{intloc}(0)) = 1$ and for every read-write integer location b such that $b \neq a$ holds $(\mathrm{Computation}(s_2))(k)(b) = (\mathrm{IExec}(I, s))(b)$ and for every finite sequence location f holds $(\mathrm{Computation}(s_2))(k)(f) = (\mathrm{IExec}(I, s))(f)$ and $\mathbf{IC}_{(\mathrm{Computation}(s_2))(k)} = \mathrm{insloc}(0)$ and for every natural number n such that $n \leq k$ holds $\mathbf{IC}_{(\mathrm{Computation}(s_2))(n)} \in \mathrm{dom}\,\mathrm{loop}\,\mathbf{if}\ a = 0\ \mathbf{then}\ \mathrm{Goto}(\mathrm{insloc}(2))\ \mathbf{else}\ (I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0)))$.

(78) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. If $s(\mathrm{intloc}(0)) = 1$ and $s(a) \leq 0$, then $\mathrm{IExec}(\mathrm{Times}(a, I), s) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = s \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(79) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s(a) > 0$. Then $(\mathrm{IExec}(I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0)), s))(a) = s(a) - 1$ and $\mathrm{IExec}(\mathrm{Times}(a, I), s) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = \mathrm{IExec}(\mathrm{Times}(a, I), \mathrm{IExec}(I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0)), s)) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(80) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, f be a finite sequence location, and a be a read-write integer location. If $s(a) \leq 0$, then $(\mathrm{IExec}(\mathrm{Times}(a, I), s))(f) = s(f)$.

(81) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, b be an integer location, and a be a read-write integer location. If $s(a) \leq 0$, then $(\mathrm{IExec}(\mathrm{Times}(a, I), s))(b) = (\mathrm{Initialize}(s))(b)$.

(82) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, f be a finite sequence location, and a be a read-write integer location. If I does not destroy a and $s(a) > 0$, then $(\mathrm{IExec}(\mathrm{Times}(a, I), s))(f) = (\mathrm{IExec}(\mathrm{Times}(a, I), \mathrm{IExec}(I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0)), s)))(f)$.

(83) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a good InitHalting macro instruction, b be an integer location, and a be a read-write integer location. If I does not destroy a and $s(a) > 0$, then $(\mathrm{IExec}(\mathrm{Times}(a, I), s))(b) = (\mathrm{IExec}(\mathrm{Times}(a, I), \mathrm{IExec}(I;\ \mathrm{SubFrom}(a, \mathrm{intloc}(0)), s)))(b)$.

Let i be an instruction of $\mathbf{SCM}_{\rm FSA}$. We say that i is good if and only if:

(Def. 6) i does not destroy $\mathrm{intloc}(0)$.

One can check that there exists an instruction of $\mathbf{SCM}_{\rm FSA}$ which is parahalting and good.

Let i be a good instruction of $\mathbf{SCM}_{\rm FSA}$ and let J be a good macro instruction. Observe that i; J is good and J; i is good.

Let i, j be good instructions of $\mathbf{SCM}_{\rm FSA}$. Observe that i; j is good.

Let a be a read-write integer location and let b be an integer location. One can verify that $a{:=}b$ is good and $\mathrm{SubFrom}(a, b)$ is good.

Let a be a read-write integer location, let b be an integer location, and let f be a finite sequence location. One can check that $a{:=}f_b$ is good.

Let a, b be integer locations and let f be a finite sequence location. Note that $f_a{:=}b$ is good.

Let a be a read-write integer location and let f be a finite sequence location. One can verify that $a{:=}\mathrm{len}\,f$ is good.

Let n be a natural number. Note that $\mathrm{intloc}(n+1)$ is read-write.

REFERENCES

[1] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\rm FSA}$. Part I. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa8a.html.

[2] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\rm FSA}$. Part II. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa8b.html.

[3] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa7b.html`.

[4] Noriko Asamoto. The `loop` and `times` macroinstruction for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 9, 1997. `http://mizar.org/JFM/Vol9/scmfsa8c.html`.

[5] 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`.

[6] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6c.html`.

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

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

[9] 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`.

[10] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **scm**. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/scm_1.html`.

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

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

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

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

[15] 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`.

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

[17] 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`.

[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. Tarski Grothendieck set theory. *Journal of Formalized Mathematics*, Axiomatics, 1989. `http://mizar.org/JFM/Axiomatics/tarski.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 and Yatsuka Nakamura. Computation in $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_3.html`.

[23] 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`.

[24] Andrzej Trybulec and Yatsuka Nakamura. Relocability for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_5.html`.

[25] 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`.

[26] 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`.

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

Received June 17, 1998

Published January 2, 2004