FORMALIZED MATHEMATICS
Volume 7, Number 1, 1998
University of Białystok

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

Jing-Chao Chen
Shanghai Jiaotong University

Yatsuka Nakamura
Shinshu University
Nagano

Summary. Up to now, many properties of macro instructions of $\mathbf{SCM}_{\rm FSA}$ 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.

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

1. The Definition of Several Notions Related to Initialization

For simplicity, we adopt the following rules: m is a natural number, I is a macro instruction, s, s_1, s_2 are states of $\mathbf{SCM}_{\rm FSA}$, 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 $\mathbf{SCM}_{\rm FSA}$ 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:

ISSN 1426–2630

(Def. 2) Initialized(I) is halting.

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

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

2. The Relationship Between Initialization Halting and Unconditional Halting

The following four propositions are true:

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

(2) For every macro instruction I and for all natural numbers i, m, n holds $\operatorname{dom} I \cap$ dom((intloc(i)$\dot{\longmapsto}m$)$+\cdot$ Start-At(insloc(n))) $= \emptyset$.

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

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

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

One can prove the following 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 $\mathbf{SCM}_{\rm FSA}$ such that Initialized(I) $\subseteq s$ holds s(intloc(0)) $= 1$.

Let us mention that every macro instruction which is paraclosed is also InitClosed.

Let us note that every macro instruction which is parahalting is also InitHalting.

One can check 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.

3. The Other Properties of Initialization Halting

One can prove the following two propositions:

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

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

Let I be a InitHalting macro instruction. Note that Initialized(I) is halting.

Let us observe that every macro instruction which is InitHalting is also non empty.

The following propositions are true:

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

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

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

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

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

Let us observe 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.

Next we state several propositions:

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

(18) Let I be a InitClosed macro instruction and J be a macro instruction. Suppose $\mathrm{Initialized}(I) \subseteq s$ and s is halting. Let given m. Suppose $m \leqslant \mathrm{LifeSpan}(s)$. 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 $s+\cdot I+\cdot((\mathrm{intloc}(i)\dot{\longmapsto}m)+\cdot \text{Start-At}(\mathrm{insloc}(n))) = (s+\cdot((\mathrm{intloc}(i)\dot{\longmapsto}m)+\cdot \text{Start-At}(\mathrm{insloc}(n))))+\cdot I$.

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

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

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

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

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

4. The Initialization Halting for Two Continuous Macro-Instructions

One can prove the following proposition

(25) Let I be a keepInt0 1 InitHalting macro instruction, J be a InitHalting macro instruction, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $\text{Initialized}(I;J) \subseteq s$. Then

(i) $\mathbf{IC}_{(\text{Computation}(s))(\text{LifeSpan}(s+\cdot I)+1)} = \mathrm{insloc}(\mathrm{card}\, I)$,

(ii) $(\text{Computation}(s))(\text{LifeSpan}(s+\cdot I)+1)\restriction(\text{Int-Locations} \cup \text{FinSeq-Locations}) = ((\text{Computation}(s+\cdot I))(\text{LifeSpan}(s+\cdot I))+\cdot \text{Initialized}(J))\restriction(\text{Int-Locations} \cup \text{FinSeq-Locations})$,

(iii) $\text{ProgramPart}(\text{Relocated}(J, \mathrm{card}\, I)) \subseteq (\text{Computation}(s))(\text{LifeSpan}(s+\cdot I)+1)$,

(iv) $(\text{Computation}(s))(\text{LifeSpan}(s+\cdot I) + 1)(\mathrm{intloc}(0)) = 1$,

(v) s is halting,

(vi) $\text{LifeSpan}(s) = \text{LifeSpan}(s+\cdot I)+1+\text{LifeSpan}(\text{Result}(s+\cdot I)+\cdot \text{Initialized}(J))$, and

(vii) if J is keeping 0, then $(\text{Result}(s))(\mathrm{intloc}(0)) = 1$.

Let I be a keepInt0 1 InitHalting macro instruction and let J be a InitHalting macro instruction. Note 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 a InitClosed macro instruction. Suppose $\mathrm{Initialized}(I;J) \subseteq s$. Let k be a natural number. Then $(\mathrm{Computation}(\mathrm{Result}(s+\cdot I)+\cdot \mathrm{Initialized}(J)))(k)$ $+\cdot$ Start-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 a 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 a InitHalting macro instruction. Then $\mathrm{IExec}(I;J, s) = \mathrm{IExec}(J, \mathrm{IExec}(I, s)) +\cdot$ Start-At($\mathbf{IC}_{\mathrm{IExec}(J,\mathrm{IExec}(I,s))} + \mathrm{card}\, I$).

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

Let I, J be keepInt0 1 macro instructions. One can verify 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. One can check 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. Observe 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. One can check that I;j is InitHalting.

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

Next we state several propositions:

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

(IExec(J, IExec(I, s)))(a).

(31) Let I be a keepInt0 1 InitHalting macro instruction and J be a InitHalting macro instruction. Then (IExec(I;J, s))(f) = (IExec(J, IExec(I, s)))(f).

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

(33) Let I be a keepInt0 1 InitHalting macro instruction and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then (IExec($I$;$j, s$))($a$) = (Exec($j$, 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 (IExec($I$;$j, s$))($f$) = (Exec($j$, 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 $\mathbf{IC}_{(\mathrm{Computation}(s+\cdot \mathrm{Initialized}(I)))(k)} \in$ dom I.

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

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

We now state three propositions:

(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 Initialized(I) $\subseteq s$. Let k be a natural number. Then (Computation(s))(k)(a) = s(a).

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

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

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

We now state several 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 a InitHalting macro instruction, and a be an integer location. Then (IExec(i;J, s))(a) = (IExec(J, Exec(i, Initialize(s))))(a).

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

(40) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$ 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 $\mathbf{SCM}_{\mathrm{FSA}}$ 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 $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{IExec}(I, s) = \mathrm{IExec}(I, \mathrm{Initialize}(s))$.

5. IF-Programs with Initialization Halting

The following propositions are true:

(43) 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 **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 $\mathbf{SCM}_{\mathrm{FSA}}$, 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** $J, s) = \mathrm{IExec}(I, s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 3))$.

(45) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be macro instructions, and a be a read-write integer location. Suppose $s(a) \neq 0$ 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 $\mathbf{SCM}_{\mathrm{FSA}}$. 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** $J, s) = \mathrm{IExec}(J, s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 3))$.

(47) 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** $J, s) = \mathrm{IExec}(I, s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 3))$ and if $s(a) \neq 0$, then IExec(**if** $a = 0$ **then** I **else** $J, s) = \mathrm{IExec}(J, s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 3))$.

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

(iii) if $s(a) \neq 0$, then for every integer location d holds (IExec(**if** $a = 0$ **then** I **else** J, s))(d) = (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).

(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) = IExec(I, s)+· Start-At(insloc(card I + 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 $s(a) \leqslant 0$ 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 $s(a) \leqslant 0$ 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)+· Start-At(insloc(card I + card J + 3)).

(53) 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) = IExec(I, s)+· Start-At(insloc(card I + card J + 3)) and if $s(a) \leqslant 0$, then IExec(**if** $a > 0$ **then** I **else** J, s) = IExec(J, s)+· Start-At(insloc(card I + card J + 3)).

(54) 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) $\mathbf{IC}_{\mathrm{IExec}(\textbf{if}\ a>0\ \textbf{then}\ I\ \textbf{else}\ J,s)}$ = insloc(card I + 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 $s(a) \leqslant 0$, 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 $\mathbf{SCM}_{\rm FSA}$, 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) = IExec(I, s)+· Start-At(insloc(card I + card J + card J + 7)).

(56) 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 J is closed onInit

s and J is halting onInit s. Then IExec(**if** $a < 0$ **then** I **else** J, s) = IExec(J, s)+· 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)+· 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)+· Start-At(insloc(card I + card J + card J + 7)), and

(iii) if $s(a) \geqslant 0$, then IExec(**if** $a < 0$ **then** I **else** J, s) = IExec(J, s)+· 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 a number of 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 a InitHalting macro instruction, and a be a read-write integer location. Then (IExec(I, s))(a) = (Computation(Initialize(s)+·(I+· Start-At(insloc(0))))) (LifeSpan(Initialize(s)+·(I+· Start-At(insloc(0)))))(a).

(62) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a 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)+·(I+· Start-At(insloc(0)))))(k)(a).

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

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

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

(67) $I \subseteq s +\cdot$ 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 $m \leqslant$ LifeSpan$(s +\cdot$ Initialized$(I))$. Then (Computation$(s +\cdot$ Initialized(I)))(m) and (Computation$(s +\cdot$ Initialized (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 $m <$ LifeSpan$(s +\cdot$ Initialized$(I))$, then CurInstr((Computation$(s +\cdot$ Initialized(I)))(m)) = CurInstr((Computation$(s +\cdot$ Initialized(loop I)))(m)).

(70) For every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds $l \notin$ dom((intloc(0)$\longmapsto$1) $+\cdot$ Start-At(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 CurInstr((Computation$(s +\cdot$ Initialized(loop I))) (LifeSpan$(s +\cdot$ Initialized(I)))) = goto insloc(0) and for every natural number m such that $m \leqslant$ LifeSpan$(s +\cdot$ Initialized$(I))$ holds CurInstr((Computation$(s +\cdot$ Initialized(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 CurInstr((Computation$(s +\cdot$ Initialized(loop I)))(LifeSpan $(s +\cdot$ 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(intloc(0)) $= 1$ and $s(a) > 0$. Then loop **if** $a = 0$ **then** Goto(insloc(2)) **else** (I;SubFrom$(a$, 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;SubFrom$(a$, intloc(0)))) is pseudo-closed

on s.

6. LOOP-Programs with Initialization Halting

We now state two propositions:

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

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

Next we state several propositions:

(77) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a good InitHalting macro instruction, and a be a read-write integer location. Suppose I does not destroy a and $s(\mathrm{intloc}(0)) = 1$ and $s(a) > 0$. Then there exists a state s_2 of $\mathbf{SCM}_{\mathrm{FSA}}$ and there exists a natural number k such that

(i) $s_2 = s +\cdot \mathrm{Initialized}(\mathrm{loop}\, \mathbf{if}\ a = 0\ \mathbf{then}\ \mathrm{Goto}(\mathrm{insloc}(2))$ $\mathbf{else}\ (I;\mathrm{SubFrom}(a, \mathrm{intloc}(0))))$,

(ii) $k = \mathrm{LifeSpan}(s +\cdot \mathrm{Initialized}(\mathbf{if}\ a = 0\ \mathbf{then}\ \mathrm{Goto}(\mathrm{insloc}(2))$ $\mathbf{else}\ (I;\mathrm{SubFrom}(a, \mathrm{intloc}(0))))) + 1$,

(iii) $(\mathrm{Computation}(s_2))(k)(a) = s(a) - 1$,

(iv) $(\mathrm{Computation}(s_2))(k)(\mathrm{intloc}(0)) = 1$,

(v) for every read-write integer location b such that $b \neq a$ holds $(\mathrm{Computation}(s_2))(k)(b) = (\mathrm{IExec}(I, s))(b)$,

(vi) for every finite sequence location f holds $(\mathrm{Computation}(s_2))(k)(f) = (\mathrm{IExec}(I, s))(f)$,

(vii) $\mathbf{IC}_{(\mathrm{Computation}(s_2))(k)} = \mathrm{insloc}(0)$, and

(viii) for every natural number n such that $n \leqslant k$ 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 $\mathbf{SCM}_{\mathrm{FSA}}$, I be a good InitHalting macro instruction, and a be a read-write integer location. If $s(\mathrm{intloc}(0)) = 1$ and $s(a) \leqslant 0$, then $\mathrm{IExec}(\mathrm{Times}(a, I), s) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$.

(79) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, 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 $\mathrm{IExec}(\mathrm{Times}(a, I), s) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = \mathrm{IExec}(\mathrm{Times}(a, I), \mathrm{IExec}(I;\mathrm{SubFrom}(a, \mathrm{intloc}(0)), s)) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$.

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

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

(82) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, 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 $\mathbf{SCM}_{\mathrm{FSA}}$, 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 $\mathbf{SCM}_{\mathrm{FSA}}$. We say that i is good if and only if:

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

Let us observe that there exists an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ which is parahalting and good.

Let i be a good instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ 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 $\mathbf{SCM}_{\mathrm{FSA}}$. Note that $i;j$ is good.

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

Let a, b be integer locations and let f be a finite sequence location. One can check 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. One can check that $\mathrm{intloc}(n + 1)$ is read-write.

References

[1] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part I. *Formalized Mathematics*, 6(**1**):65–72, 1997.

[2] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):73–80, 1997.

[3] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):59–63, 1997.

[4] Noriko Asamoto. The `loop` and `Times` macroinstruction for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**4**):483–497, 1997.

[5] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Formalized Mathematics*, 6(**1**):41–47, 1997.
[6] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Formalized Mathematics*, 6(**1**):53–57, 1997.
[7] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.
[8] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
[9] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.
[10] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **scm**. *Formalized Mathematics*, 4(**1**):61–67, 1993.
[11] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.
[12] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.
[13] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[14] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[15] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[16] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[17] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[18] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[19] Andrzej Trybulec and Yatsuka Nakamura. Computation in $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):537–542, 1996.
[20] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[21] Andrzej Trybulec and Yatsuka Nakamura. Relocability for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):583–586, 1996.
[22] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
[23] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[24] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[25] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[26] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[27] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received June 17, 1998
