FORMALIZED MATHEMATICS
Volume 6, Number 4, 1997
University of Białystok

While Macro Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$

Jing-Chao Chen[1]
Shanghai Jiaotong University
Shanghai

Summary. The article defines *while macro instructions* based on $\mathbf{SCM}_{\mathrm{FSA}}$. Some theorems about the generalized halting problems of *while macro instructions* are proved.

MML Identifier: `SCMFSA_9`.

The notation and terminology used in this paper are introduced in the following papers: [24], [32], [19], [8], [13], [33], [15], [16], [17], [12], [34], [7], [10], [14], [31], [18], [9], [20], [21], [25], [11], [23], [30], [29], [26], [27], [1], [28], [22], [5], [6], [4], [2], and [3].

The following propositions are true:

(1) For every macro instruction I and for every integer location a holds $\mathrm{card}\, if = 0(a, I; \mathrm{Goto}(\mathrm{insloc}(0)), \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}) = \mathrm{card}\, I + 6$.

(2) For every macro instruction I and for every integer location a holds $\mathrm{card}\, if > 0(a, I; \mathrm{Goto}(\mathrm{insloc}(0)), \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}) = \mathrm{card}\, I + 6$.

Let a be an integer location and let I be a macro instruction. The functor $while = 0(a, I)$ yielding a macro instruction is defined as follows:

(Def. 1) $while = 0(a, I) = if = 0(a, I; \mathrm{Goto}(\mathrm{insloc}(0)), \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}) {+\cdot} (\mathrm{insloc}(\mathrm{card}\, I + 4) \dot{\longmapsto} \mathrm{goto}\ \mathrm{insloc}(0))$.

The functor $while > 0(a, I)$ yielding a macro instruction is defined by:

(Def. 2) $while > 0(a, I) = if > 0(a, I; \mathrm{Goto}(\mathrm{insloc}(0)), \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}) {+\cdot} (\mathrm{insloc}(\mathrm{card}\, I + 4) \dot{\longmapsto} \mathrm{goto}\ \mathrm{insloc}(0))$.

The following proposition is true

[1]Part of the work was done while the author was visiting the Institute of Mathematics at the University of Białystok.

ISSN 1426–2630

(3) For every macro instruction I and for every integer location a holds $\operatorname{card} if = 0(a, \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, if > 0(a, \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, I; \mathrm{Goto}(\mathrm{insloc}(0)))) = \operatorname{card} I + 11$.

Let a be an integer location and let I be a macro instruction. The functor $while < 0(a, I)$ yields a macro instruction and is defined as follows:

(Def. 3) $while < 0(a, I) = if = 0(a, \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, if > 0(a, \mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, I; \mathrm{Goto}(\mathrm{insloc}(0)))) {+\cdot} (\mathrm{insloc}(\operatorname{card} I + 4) \longmapsto \mathrm{goto}\ \mathrm{insloc}(0))$.

Next we state a number of propositions:

(4) For every macro instruction I and for every integer location a holds $\operatorname{card} while = 0(a, I) = \operatorname{card} I + 6$.

(5) For every macro instruction I and for every integer location a holds $\operatorname{card} while > 0(a, I) = \operatorname{card} I + 6$.

(6) For every macro instruction I and for every integer location a holds $\operatorname{card} while < 0(a, I) = \operatorname{card} I + 11$.

(7) For every integer location a and for every instruction-location l of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathbf{if}\ a = 0\ \mathbf{goto}\ l \neq \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(8) For every integer location a and for every instruction-location l of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathbf{if}\ a > 0\ \mathbf{goto}\ l \neq \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(9) For every instruction-location l of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{goto}\ l \neq \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(10) Let a be an integer location and I be a macro instruction. Then $\mathrm{insloc}(0) \in \operatorname{dom} while = 0(a, I)$ and $\mathrm{insloc}(1) \in \operatorname{dom} while = 0(a, I)$ and $\mathrm{insloc}(0) \in \operatorname{dom} while > 0(a, I)$ and $\mathrm{insloc}(1) \in \operatorname{dom} while > 0(a, I)$.

(11) Let a be an integer location and I be a macro instruction. Then $(while = 0(a, I))(\mathrm{insloc}(0)) = \mathbf{if}\ a = 0\ \mathbf{goto}\ \mathrm{insloc}(4)$ and $(while = 0(a, I))(\mathrm{insloc}(1)) = \mathrm{goto}\ \mathrm{insloc}(2)$ and $(while > 0(a, I))(\mathrm{insloc}(0)) = \mathbf{if}\ a > 0\ \mathbf{goto}\ \mathrm{insloc}(4)$ and $(while > 0(a, I))(\mathrm{insloc}(1)) = \mathrm{goto}\ \mathrm{insloc}(2)$.

(12) Let a be an integer location, I be a macro instruction, and k be a natural number. If $k < 6$, then $\mathrm{insloc}(k) \in \operatorname{dom} while = 0(a, I)$.

(13) Let a be an integer location, I be a macro instruction, and k be a natural number. If $k < 6$, then $\mathrm{insloc}(\operatorname{card} I + k) \in \operatorname{dom} while = 0(a, I)$.

(14) For every integer location a and for every macro instruction I holds $(while = 0(a, I))(\mathrm{insloc}(\operatorname{card} I + 5)) = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(15) For every integer location a and for every macro instruction I holds $(while = 0(a, I))(\mathrm{insloc}(3)) = \mathrm{goto}\ \mathrm{insloc}(\operatorname{card} I + 5)$.

(16) For every integer location a and for every macro instruction I holds $(while = 0(a, I))(\mathrm{insloc}(2)) = \mathrm{goto}\ \mathrm{insloc}(3)$.

(17) Let a be an integer location, I be a macro instruction, and k be a natural number. If $k < \operatorname{card} I + 6$, then $\mathrm{insloc}(k) \in \operatorname{dom} while = 0(a, I)$.

(18) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. If $s(a) \neq 0$, then $\mathit{while} = 0(a, I)$ is halting on s and $\mathit{while} = 0(a, I)$ is closed on s.

(19) Let a be an integer location, I be a macro instruction, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k be a natural number. Suppose that

(i) I is closed on s and halting on s,

(ii) $k < \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0))))$,

(iii) $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}=0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1+k)} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(k)} + 4$, and

(iv) $(\mathrm{Computation}(s{+\cdot}(\mathit{while} = 0(a, I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1 + k) \restriction(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = (\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(k)\restriction(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

Then $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}=0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1+k+1)} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(k+1)} + 4$ and $(\mathrm{Computation}(s{+\cdot}(\mathit{while} = 0(a, I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1 + k + 1)\restriction(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = (\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(k + 1)\restriction(\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(20) Let a be an integer location, I be a macro instruction, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose I is closed on s and halting on s and $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}=0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1+\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))} + 4$. Then $\mathrm{CurInstr}((\mathrm{Computation}(s{+\cdot}(\mathit{while} = 0(a, I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(1 + \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))) = \mathbf{goto}\ \mathrm{insloc}(\mathrm{card}\, I + 4)$.

(21) For every integer location a and for every macro instruction I holds $(\mathit{while} = 0(a, I))(\mathrm{insloc}(\mathrm{card}\, I + 4)) = \mathbf{goto}\ \mathrm{insloc}(0)$.

(22) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. Suppose I is closed on s and halting on s and $s(a) = 0$. Then $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}=0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0))))+3)} = \mathrm{insloc}(0)$ and for every natural number k such that $k \leqslant \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))) + 3$ holds $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}=0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(k)} \in \mathrm{dom}\, \mathit{while} = 0(a, I)$.

In the sequel s denotes a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I denotes a macro instruction, and a denotes a read-write integer location.

Let us consider s, I, a. The functor $\mathit{StepWhile} = 0(a, I, s)$ yields a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) and is defined by the conditions (Def. 4).

(Def. 4)(i) $(\mathit{StepWhile} = 0(a, I, s))(0) = s$, and

(ii) for every natural number i and for every element x of $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) such that $x = (\mathit{StepWhile} = 0(a, I, s))(i)$

holds $(StepWhile = 0(a, I, s))(i + 1) = (\mathrm{Computation}(x{+\cdot}(while = 0(a, I){+\cdot}s_0)))(\mathrm{LifeSpan}(x{+\cdot}(I{+\cdot}s_0)) + 3)$.

In the sequel k, n are natural numbers.

We now state three propositions:

(23) $(StepWhile = 0(a, I, s))(0) = s$.

(24) $(StepWhile = 0(a, I, s))(k + 1) = (\mathrm{Computation}((StepWhile = 0(a, I, s))(k){+\cdot}(while = 0(a, I){+\cdot}s_0)))(\mathrm{LifeSpan}((StepWhile = 0(a, I, s))(k){+\cdot}(I{+\cdot}s_0)) + 3)$.

(25) $(StepWhile = 0(a, I, s))(k + 1) = (StepWhile = 0(a, I, (StepWhile = 0(a, I, s))(k)))(1)$.

The scheme *MinIndex* deals with a unary functor $\mathcal{F}$ yielding a natural number and a natural number $\mathcal{A}$, and states that:

> There exists k such that $\mathcal{F}(k) = 0$ and for every n such that $\mathcal{F}(n) = 0$ holds $k \leqslant n$

provided the parameters meet the following conditions:

- $\mathcal{F}(0) = \mathcal{A}$, and
- For every k holds $\mathcal{F}(k + 1) < \mathcal{F}(k)$ or $\mathcal{F}(k) = 0$.

We now state a number of propositions:

(26) For all functions f, g holds $f{+\cdot}g{+\cdot}g = f{+\cdot}g$.

(27) For all functions f, g, h and for every set D such that $(f{+\cdot}g){\restriction}D = h{\restriction}D$ holds $(h{+\cdot}g){\restriction}D = (f{+\cdot}g){\restriction}D$.

(28) For all functions f, g, h and for every set D such that $f{\restriction}D = h{\restriction}D$ holds $(h{+\cdot}g){\restriction}D = (f{+\cdot}g){\restriction}D$.

(29) For all states s_1, s_2 of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and $s_1{\restriction}(\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2{\restriction}(\text{Int-Locations} \cup \text{FinSeq-Locations})$ and $s_1{\restriction}I_1 = s_2{\restriction}I_1$ holds $s_1 = s_2$.

(30) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(StepWhile = 0(a, I, s))(0 + 1) = (\mathrm{Computation}(s{+\cdot}(while = 0(a, I){+\cdot}s_0)))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}s_0)) + 3)$.

(31) Let I be a macro instruction, a be a read-write integer location, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k, n be natural numbers. Suppose $\mathbf{IC}_{(StepWhile=0(a,I,s))(k)} = \mathrm{insloc}(0)$ and $(StepWhile = 0(a, I, s))(k) = (\mathrm{Computation}(s{+\cdot}(while = 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(n)$. Then $(StepWhile = 0(a, I, s))(k) = (StepWhile = 0(a, I, s))(k){+\cdot}(while = 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))$ and $(StepWhile = 0(a, I, s))(k + 1) = (\mathrm{Computation}(s{+\cdot}(while = 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(n + (\mathrm{LifeSpan}((StepWhile = 0(a, I, s))(k){+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))) + 3))$.

(32) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that

(i) for every natural number k holds I is closed on $(StepWhile = 0(a, I, s))(k)$ and halting on $(StepWhile = 0(a, I, s))(k)$, and

(ii) there exists a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that for every natural number k holds $f((StepWhile = 0(a, I, s))(k + 1)) < f((StepWhile = 0(a, I, s))(k))$ or $f((StepWhile = 0(a, I, s))(k)) = 0$ but $f((StepWhile = 0(a, I, s))(k)) = 0$ iff $(StepWhile = 0(a, I, s))(k)(a) \neq 0$.

Then $while = 0(a, I)$ is halting on s and $while = 0(a, I)$ is closed on s.

(33) Let I be a parahalting macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that let k be a natural number. Then $f((StepWhile = 0(a, I, s))(k + 1)) < f((StepWhile = 0(a, I, s))(k))$ or $f((StepWhile = 0(a, I, s))(k)) = 0$ but $f((StepWhile = 0(a, I, s))(k)) = 0$ iff $(StepWhile = 0(a, I, s))(k)(a) \neq 0$. Then $while = 0(a, I)$ is halting on s and $while = 0(a, I)$ is closed on s.

(34) Let I be a parahalting macro instruction and a be a read-write integer location. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $f((StepWhile = 0(a, I, s))(1)) < f(s)$ or $f(s) = 0$ but $f(s) = 0$ iff $s(a) \neq 0$. Then $while = 0(a, I)$ is parahalting.

(35) For all instructions-locations l_1, l_2 of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every integer location a holds $l_1 \longmapsto$goto l_2 does not destroy a.

(36) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ such that i does not destroy intloc(0) holds Macro(i) is good.

Let I, J be good macro instructions and let a be an integer location. Note that $if = 0(a, I, J)$ is good.

Let I be a good macro instruction and let a be an integer location. One can verify that $while = 0(a, I)$ is good.

We now state a number of propositions:

(37) Let a be an integer location, I be a macro instruction, and k be a natural number. If $k < 6$, then insloc$(k) \in \mathrm{dom}\, while > 0(a, I)$.

(38) Let a be an integer location, I be a macro instruction, and k be a natural number. If $k < 6$, then insloc$(\mathrm{card}\, I + k) \in \mathrm{dom}\, while > 0(a, I)$.

(39) For every integer location a and for every macro instruction I holds $(while > 0(a, I))(\mathrm{insloc}(\mathrm{card}\, I + 5)) = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(40) For every integer location a and for every macro instruction I holds $(while > 0(a, I))(\mathrm{insloc}(3)) = \mathrm{goto}\ \mathrm{insloc}(\mathrm{card}\, I + 5)$.

(41) For every integer location a and for every macro instruction I holds $(while > 0(a, I))(\mathrm{insloc}(2)) = \mathrm{goto}\ \mathrm{insloc}(3)$.

(42) Let a be an integer location, I be a macro instruction, and k be a natural

number. If $k < \operatorname{card} I + 6$, then $\mathrm{insloc}(k) \in \operatorname{dom} \mathit{while} > 0(a, I)$.

(43) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. If $s(a) \leqslant 0$, then $\mathit{while} > 0(a, I)$ is halting on s and $\mathit{while} > 0(a, I)$ is closed on s.

(44) Let a be an integer location, I be a macro instruction, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k be a natural number. Suppose that

(i) I is closed on s and halting on s,

(ii) $k < \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0))))$,

(iii) $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}>0(a,I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1+k)} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(k)} + 4$, and

(iv) $(\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1+k){\restriction}D = (\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(k){\restriction}D$.

Then $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}>0(a,I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1+k+1)} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(k+1)}+4$ and $(\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1 + k + 1){\restriction}D = (\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(k + 1){\restriction}D$.

(45) Let a be an integer location, I be a macro instruction, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose I is closed on s and halting on s and $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}>0(a,I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1+\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))} = \mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))} + 4$. Then $\mathrm{CurInstr}((\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a, I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(1 + \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))) = \mathrm{goto}\ \mathrm{insloc}(\operatorname{card} I + 4)$.

(46) For every integer location a and for every macro instruction I holds $(\mathit{while} > 0(a, I))(\mathrm{insloc}(\operatorname{card} I + 4)) = \mathrm{goto}\ \mathrm{insloc}(0)$.

(47) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. Suppose I is closed on s and halting on s and $s(a) > 0$.

Then $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}>0(a,I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0))))+3)} = \mathrm{insloc}(0)$ and for every natural number k such that $k \leqslant \mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))) + 3$ holds $\mathbf{IC}_{(\mathrm{Computation}(s{+\cdot}(\mathit{while}>0(a,I){+\cdot}\,\text{Start-At}(\mathrm{insloc}(0)))))(k)} \in \operatorname{dom} \mathit{while} > 0(a, I)$.

In the sequel s denotes a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I denotes a macro instruction, and a denotes a read-write integer location.

Let us consider s, I, a. The functor $\mathit{StepWhile} > 0(a, I, s)$ yielding a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) is defined by the conditions (Def. 5).

(Def. 5)(i) $(\mathit{StepWhile} > 0(a, I, s))(0) = s$, and

(ii) for every natural number i and for every element x of $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) such that $x = (\mathit{StepWhile} > 0(a, I, s))(i)$ holds $(\mathit{StepWhile} > 0(a, I, s))(i + 1) = (\mathrm{Computation}(x{+\cdot}(\mathit{while} > 0(a, I){+\cdot}s_0)))(\mathrm{LifeSpan}(x{+\cdot}(I{+\cdot}s_0)) + 3)$.

One can prove the following propositions:

(48) $(\mathit{StepWhile} > 0(a,I,s))(0) = s$.

(49) $(\mathit{StepWhile} > 0(a,I,s))(k+1) = (\mathrm{Computation}((\mathit{StepWhile} > 0(a,I,s))(k){+\cdot}(\mathit{while} > 0(a,I){+\cdot}s_0)))(\mathrm{LifeSpan}((\mathit{StepWhile} > 0(a,I,s))(k){+\cdot}(I{+\cdot}s_0)) + 3)$.

(50) $(\mathit{StepWhile} > 0(a,I,s))(k+1) = (\mathit{StepWhile} > 0(a,I,(\mathit{StepWhile} > 0(a,I,s))(k)))(1)$.

(51) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathit{StepWhile} > 0(a,I,s))(0+1) = (\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a,I){+\cdot}s_0)))(\mathrm{LifeSpan}(s{+\cdot}(I{+\cdot}s_0)) + 3)$.

(52) Let I be a macro instruction, a be a read-write integer location, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k, n be natural numbers. Suppose $\mathbf{IC}_{(\mathit{StepWhile} > 0(a,I,s))(k)} = \mathrm{insloc}(0)$ and $(\mathit{StepWhile} > 0(a,I,s))(k) = (\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(n)$. Then $(\mathit{StepWhile} > 0(a,I,s))(k) = (\mathit{StepWhile} > 0(a,I,s))(k){+\cdot}(\mathit{while} > 0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))$ and $(\mathit{StepWhile} > 0(a,I,s))(k+1) = (\mathrm{Computation}(s{+\cdot}(\mathit{while} > 0(a,I){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(n + (\mathrm{LifeSpan}((\mathit{StepWhile} > 0(a,I,s))(k){+\cdot}(I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))) + 3))$.

(53) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that

(i) for every natural number k holds I is closed on $(\mathit{StepWhile} > 0(a,I,s))(k)$ and halting on $(\mathit{StepWhile} > 0(a,I,s))(k)$, and

(ii) there exists a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that for every natural number k holds $f((\mathit{StepWhile} > 0(a,I,s))(k+1)) < f((\mathit{StepWhile} > 0(a,I,s))(k))$ or $f((\mathit{StepWhile} > 0(a,I,s))(k)) = 0$ but $f((\mathit{StepWhile} > 0(a,I,s))(k)) = 0$ iff $(\mathit{StepWhile} > 0(a,I,s))(k)(a) \leqslant 0$.

Then $\mathit{while} > 0(a,I)$ is halting on s and $\mathit{while} > 0(a,I)$ is closed on s.

(54) Let I be a parahalting macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that let k be a natural number. Then $f((\mathit{StepWhile} > 0(a,I,s))(k+1)) < f((\mathit{StepWhile} > 0(a,I,s))(k))$ or $f((\mathit{StepWhile} > 0(a,I,s))(k)) = 0$ but $f((\mathit{StepWhile} > 0(a,I,s))(k)) = 0$ iff $(\mathit{StepWhile} > 0(a,I,s))(k)(a) \leqslant 0$. Then $\mathit{while} > 0(a,I)$ is halting on s and $\mathit{while} > 0(a,I)$ is closed on s.

(55) Let I be a parahalting macro instruction and a be a read-write integer location. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $f((\mathit{StepWhile} > 0(a,I,s))(1)) < f(s)$ or $f(s) = 0$ but $f(s) = 0$ iff $s(a) \leqslant 0$. Then $\mathit{while} > 0(a,I)$ is parahalting.

Let I, J be good macro instructions and let a be an integer location. One can verify that $if > 0(a, I, J)$ is good.

Let I be a good macro instruction and let a be an integer location. One can verify that $while > 0(a, I)$ is good.

Acknowledgments

The author wishes to thank Prof. Andrzej Trybulec and Dr. Grzegorz Bancerek for their helpful comments and encouragement during his stay in Białystok.

References

[1] Noriko Asamoto. Some multi instructions defined by sequence of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):615–619, 1996.
[2] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part I. *Formalized Mathematics*, 6(**1**):65–72, 1997.
[3] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):73–80, 1997.
[4] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):59–63, 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 Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[11] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **scm**. *Formalized Mathematics*, 4(**1**):61–67, 1993.
[12] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.
[13] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.
[14] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.
[15] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[16] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[17] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Formalized Mathematics*, 1(**3**):521–527, 1990.
[18] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.
[19] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[20] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[21] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[22] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[23] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.

[24] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[25] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[26] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[27] Andrzej Trybulec and Yatsuka Nakamura. Relocability for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):583–586, 1996.
[28] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
[29] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[30] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **scm**. *Formalized Mathematics*, 5(**4**):507–512, 1996.
[31] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[32] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[33] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[34] Wojciech Zielonka. Preliminaries to the Lambek calculus. *Formalized Mathematics*, 2(**3**):413–418, 1991.

Received December 10, 1997