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

Conditional Branch Macro Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II

Noriko Asamoto
Ochanomizu University
Tokyo

MML Identifier: SCMFSA8B.

WWW: http://mizar.org/JFM/Vol8/scmfsa8b.html

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

The following propositions are true:

(1) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}} \in \mathrm{dom}\, s$.

(2) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every instruction-location l of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $l \in \mathrm{dom}\, s$.

(3) For every macro instruction I and for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ such that I is closed on s holds $\mathrm{insloc}(0) \in \mathrm{dom}\, I$.

(4) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for all instruction-locations l_1, l_2 of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $s {+\cdot}\, \text{Start-At}(l_1) {+\cdot}\, \text{Start-At}(l_2) = s {+\cdot}\, \text{Start-At}(l_2)$.

(5) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every macro instruction I holds $\mathrm{Initialize}(s) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = (s {+\cdot}\, \mathrm{Initialized}(I)) \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$.

(6) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$ and I be a macro instruction. If $s_1 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$, then if I is closed on s_1, then I is closed on s_2.

(7) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$ and I, J be macro instructions. Suppose $s_1 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$. Then $s_1 {+\cdot} (I {+\cdot}\, \text{Start-At}(\mathrm{insloc}(0)))$ and $s_2 {+\cdot} (J {+\cdot}\, \text{Start-At}(\mathrm{insloc}(0)))$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(8) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$ and I be a macro instruction. Suppose $s_1 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2 \restriction (\text{Int-Locations} \cup \text{FinSeq-Locations})$. Suppose I is closed on s_1 and halting on s_1. Then I is closed on s_2 and halting on s_2.

(9) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for all macro instructions I, J holds I is closed on $\mathrm{Initialize}(s)$ iff I is closed on $s {+\cdot}\, \mathrm{Initialized}(J)$.

(10) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J be macro instructions, and l be an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$. Then I is closed on s if and only if I is closed on $s {+\cdot} (I {+\cdot}\, \text{Start-At}(l))$.

(11) Let s_1, s_2 be states of $\mathbf{SCM}_{\rm FSA}$ and I be a macro instruction. Suppose $I +\cdot$ Start-At(insloc(0)) $\subseteq s_1$ and I is closed on s_1. Let n be a natural number. Suppose ProgramPart(Relocated(I,n)) $\subseteq s_2$ and $\mathbf{IC}_{(s_2)} =$ insloc(n) and $s_1 \restriction$(Int-Locations $\cup$ FinSeq-Locations) $= s_2 \restriction$(Int-Locations $\cup$ FinSeq-Locations). Let i be a natural number. Then $\mathbf{IC}_{({\rm Computation}(s_1))(i)} + n = \mathbf{IC}_{({\rm Computation}(s_2))(i)}$ and IncAddr(CurInstr((Computation(s_1))(i)), n) $=$ CurInstr((Computation(s_2))(i)) and (Computation(s_1))(i)$\restriction$(Int-Locations $\cup$ FinSeq-Locations) $=$ (Computation(s_2))(i)$\restriction$(Int-Locations $\cup$ FinSeq-Locations).

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

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

Let a be an integer location and let I, J be macro instructions. The functor **if** $a = 0$ **then** I **else** J yields a macro instruction and is defined by:

(Def. 1) **if** $a = 0$ **then** I **else** $J =$ (**if** $a = 0$ **goto** insloc(card$J + 3$)); J; Goto(insloc(card$I + 1$)); I; $\mathrm{Stop}_{\mathbf{SCM}_{\rm FSA}}$.

The functor **if** $a > 0$ **then** I **else** J yielding a macro instruction is defined as follows:

(Def. 2) **if** $a > 0$ **then** I **else** $J =$ (**if** $a > 0$ **goto** insloc(card$J + 3$)); J; Goto(insloc(card$I + 1$)); I; $\mathrm{Stop}_{\mathbf{SCM}_{\rm FSA}}$.

Let a be an integer location and let I, J be macro instructions. The functor **if** $a < 0$ **then** I **else** J yielding a macro instruction is defined by:

(Def. 3) **if** $a < 0$ **then** I **else** $J =$ **if** $a = 0$ **then** J **else** (**if** $a > 0$ **then** J **else** I).

One can prove the following propositions:

(14) For all macro instructions I, J and for every integer location a holds card(**if** $a = 0$ **then** I **else** J) $=$ card$I$ + card$J + 4$.

(15) For all macro instructions I, J and for every integer location a holds card(**if** $a > 0$ **then** I **else** J) $=$ card$I$ + card$J + 4$.

(16) 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 on s and halting on s. Then **if** $a = 0$ **then** I **else** J is closed on s and **if** $a = 0$ **then** I **else** J is halting on s.

(17) 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 on Initialize(s) and halting on Initialize(s). Then IExec(**if** $a = 0$ **then** I **else** J, s) $=$ IExec(I,s) $+\cdot$ Start-At(insloc(card$I$ + card$J + 3$)).

(18) 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 on s and halting on s. Then **if** $a = 0$ **then** I **else** J is closed on s and **if** $a = 0$ **then** I **else** J is halting on s.

(19) 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 on Initialize(s) and halting on Initialize(s). Then IExec(**if** $a = 0$ **then** I **else** J, s) $=$ IExec(J,s) $+\cdot$ Start-At(insloc(card$I$ + card$J + 3$)).

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

(21) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be parahalting 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 $(\mathrm{IExec}(\mathbf{if}\ a = 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\mathrm{IExec}(I,s))(d)$ and for every finite sequence location f holds $(\mathrm{IExec}(\mathbf{if}\ a = 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\mathrm{IExec}(I,s))(f)$, and

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

(22) 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 on s and halting on s. Then **if** $a > 0$ **then** I **else** J is closed on s and **if** $a > 0$ **then** I **else** J is halting on s.

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

(24) 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 on s and halting on s. Then **if** $a > 0$ **then** I **else** J is closed on s and **if** $a > 0$ **then** I **else** J is halting on s.

(25) 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 on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$. Then $\mathrm{IExec}(\mathbf{if}\ a > 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s) = \mathrm{IExec}(J,s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 3))$.

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

(27) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I, J be parahalting 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 $(\mathrm{IExec}(\mathbf{if}\ a > 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\mathrm{IExec}(I,s))(d)$ and for every finite sequence location f holds $(\mathrm{IExec}(\mathbf{if}\ a > 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\mathrm{IExec}(I,s))(f)$, and

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

(28) 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 on s and halting on s. Then **if** $a < 0$ **then** I **else** J is closed on s and **if** $a < 0$ **then** I **else** J is halting on s.

(29) 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 on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$. Then $\mathrm{IExec}(\mathbf{if}\ a < 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s) = \mathrm{IExec}(I,s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + \mathrm{card}\, J + 7))$.

(30) 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 on s and halting on s. Then **if** $a < 0$ **then** I **else** J is closed on s and **if** $a < 0$ **then** I **else** J is halting on s.

(31) 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 on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$. Then $\mathrm{IExec}(\mathbf{if}\ a < 0\ \mathbf{then}\ I\ \mathbf{else}\ J, s) = \mathrm{IExec}(J,s) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + \mathrm{card}\, J + 7))$.

(32) 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 on s and halting on s. Then **if** $a < 0$ **then** I **else** J is closed on s and **if** $a < 0$ **then** I **else** J is halting on s.

(33) 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 on Initialize(s) and halting on Initialize(s). Then IExec(**if** $a < 0$ **then** I **else** $J, s) = \mathrm{IExec}(J, s) +\cdot \mathrm{Start\text{-}At}(\mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + \mathrm{card}\, J + 7))$.

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

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

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

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

Let I, J be parahalting macro instructions and let a be a read-write integer location. One can verify that **if** $a = 0$ **then** I **else** J is parahalting and **if** $a > 0$ **then** I **else** J is parahalting.

Let a, b be integer locations and let I, J be macro instructions. The functor **if** $a = b$ **then** I **else** J yields a macro instruction and is defined by:

(Def. 4) **if** $a = b$ **then** I **else** $J = \mathrm{SubFrom}(a, b)$; (**if** $a = 0$ **then** I **else** J).

The functor **if** $a > b$ **then** I **else** J yielding a macro instruction is defined by:

(Def. 5) **if** $a > b$ **then** I **else** $J = \mathrm{SubFrom}(a, b)$; (**if** $a > 0$ **then** I **else** J).

We introduce **if** $b < a$ **then** I **else** J as a synonym of **if** $a > b$ **then** I **else** J.

Let I, J be parahalting macro instructions and let a, b be read-write integer locations. One can check that **if** $a = b$ **then** I **else** J is parahalting and **if** $a > b$ **then** I **else** J is parahalting.

We now state several propositions:

(35) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every macro instruction I holds $\mathrm{Result}(s +\cdot \mathrm{Initialized}(I)) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = \mathrm{IExec}(I, s) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(36) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be an integer location. Then $\mathrm{Result}(s +\cdot \mathrm{Initialized}(I))$ and $\mathrm{IExec}(I, s)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(37) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$, i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, and a be an integer location. Suppose that

(i) for every integer location b such that $a \neq b$ holds $s_1(b) = s_2(b)$,

(ii) for every finite sequence location f holds $s_1(f) = s_2(f)$,

(iii) i does not refer a, and

(iv) $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$.

Then

(v) for every integer location b such that $a \neq b$ holds $(\mathrm{Exec}(i, s_1))(b) = (\mathrm{Exec}(i, s_2))(b)$,

(vi) for every finite sequence location f holds $(\mathrm{Exec}(i, s_1))(f) = (\mathrm{Exec}(i, s_2))(f)$, and

(vii) $\mathbf{IC}_{\mathrm{Exec}(i, s_1)} = \mathbf{IC}_{\mathrm{Exec}(i, s_2)}$.

(38) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be an integer location. Suppose that

(i) I does not refer a,

(ii) for every integer location b such that $a \neq b$ holds $s_1(b) = s_2(b)$,

(iii) for every finite sequence location f holds $s_1(f) = s_2(f)$, and

(iv) I is closed on s_1 and halting on s_1.

Let k be a natural number. Then

(v) for every integer location b such that $a \neq b$ holds $(\text{Computation}(s_1 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)(b) = (\text{Computation}(s_2 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)(b)$,

(vi) for every finite sequence location f holds $(\text{Computation}(s_1 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)(f) = (\text{Computation}(s_2 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)(f)$,

(vii) $\mathbf{IC}_{(\text{Computation}(s_1 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)} = \mathbf{IC}_{(\text{Computation}(s_2 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)}$, and

(viii) $\text{CurInstr}((\text{Computation}(s_1 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k)) = \text{CurInstr}((\text{Computation}(s_2 + \cdot (I + \cdot \text{Start-At}(\text{insloc}(0)))))(k))$.

(39) Let s be a state of $\mathbf{SCM}_{\text{FSA}}$, I, J be macro instructions, and l be an instruction-location of $\mathbf{SCM}_{\text{FSA}}$. Then I is closed on s and halting on s if and only if I is closed on $s + \cdot (I + \cdot \text{Start-At}(l))$ and halting on $s + \cdot (I + \cdot \text{Start-At}(l))$.

(40) Let s_1, s_2 be states of $\mathbf{SCM}_{\text{FSA}}$, I be a macro instruction, and a be an integer location. Suppose that

(i) I does not refer a,

(ii) for every integer location b such that $a \neq b$ holds $s_1(b) = s_2(b)$,

(iii) for every finite sequence location f holds $s_1(f) = s_2(f)$, and

(iv) I is closed on s_1 and halting on s_1.

Then I is closed on s_2 and halting on s_2.

(41) Let s_1, s_2 be states of $\mathbf{SCM}_{\text{FSA}}$, I be a macro instruction, and a be an integer location. Suppose that

(i) for every read-write integer location d such that $a \neq d$ holds $s_1(d) = s_2(d)$,

(ii) for every finite sequence location f holds $s_1(f) = s_2(f)$,

(iii) I does not refer a, and

(iv) I is closed on $\text{Initialize}(s_1)$ and halting on $\text{Initialize}(s_1)$.

Then

(v) for every integer location d such that $a \neq d$ holds $(\text{IExec}(I, s_1))(d) = (\text{IExec}(I, s_2))(d)$,

(vi) for every finite sequence location f holds $(\text{IExec}(I, s_1))(f) = (\text{IExec}(I, s_2))(f)$, and

(vii) $\mathbf{IC}_{\text{IExec}(I, s_1)} = \mathbf{IC}_{\text{IExec}(I, s_2)}$.

(42) Let s be a state of $\mathbf{SCM}_{\text{FSA}}$, I, J be parahalting macro instructions, and a, b be read-write integer locations. Suppose I does not refer a and J does not refer a. Then

(i) $\mathbf{IC}_{\text{IExec}(\mathbf{if}\ a = b\ \mathbf{then}\ I\ \mathbf{else}\ J, s)} = \text{insloc}(\text{card}\, I + \text{card}\, J + 5)$,

(ii) if $s(a) = s(b)$, then for every integer location d such that $a \neq d$ holds $(\text{IExec}(\mathbf{if}\ a = b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\text{IExec}(I, s))(d)$ and for every finite sequence location f holds $(\text{IExec}(\mathbf{if}\ a = b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\text{IExec}(I, s))(f)$, and

(iii) if $s(a) \neq s(b)$, then for every integer location d such that $a \neq d$ holds $(\text{IExec}(\mathbf{if}\ a = b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\text{IExec}(J, s))(d)$ and for every finite sequence location f holds $(\text{IExec}(\mathbf{if}\ a = b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\text{IExec}(J, s))(f)$.

(43) Let s be a state of $\mathbf{SCM}_{\text{FSA}}$, I, J be parahalting macro instructions, and a, b be read-write integer locations. Suppose I does not refer a and J does not refer a. Then

(i) $\mathbf{IC}_{\text{IExec}(\mathbf{if}\ a > b\ \mathbf{then}\ I\ \mathbf{else}\ J, s)} = \text{insloc}(\text{card}\, I + \text{card}\, J + 5)$,

(ii) if $s(a) > s(b)$, then for every integer location d such that $a \neq d$ holds $(\text{IExec}(\mathbf{if}\ a > b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\text{IExec}(I, s))(d)$ and for every finite sequence location f holds $(\text{IExec}(\mathbf{if}\ a > b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\text{IExec}(I, s))(f)$, and

(iii) if $s(a) \leq s(b)$, then for every integer location d such that $a \neq d$ holds $(\text{IExec}(\mathbf{if}\ a > b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(d) = (\text{IExec}(J, s))(d)$ and for every finite sequence location f holds $(\text{IExec}(\mathbf{if}\ a > b\ \mathbf{then}\ I\ \mathbf{else}\ J, s))(f) = (\text{IExec}(J, s))(f)$.

REFERENCES

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

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

[3] 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`.

[4] 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`.

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

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

[7] 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`.

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

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

[10] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/finseq_2.html`.

[11] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/funct_4.html`.

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

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

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

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

[16] 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`.

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

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

[19] 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`.

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

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

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

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

Received August 27, 1996

Published January 2, 2004