FORMALIZED MATHEMATICS
Volume 6, Number 1, 1997
Warsaw University - Białystok

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

Noriko Asamoto
Ochanomizu University
Tokyo

MML Identifier: SCMFSA8B.

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

One can prove the following propositions:

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

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

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

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

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

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

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

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

ISSN 1426–2630

(Int-Locations $\cup$ 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 Initialize(s) iff I is closed on $s +\cdot$ Initialized(J).

(10) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let 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 $\mathbf{SCM}_{\mathrm{FSA}}$ and let I be a macro instruction. Suppose $I +\cdot \text{Start-At}(\text{insloc}(0)) \subseteq s_1$ and I is closed on s_1. Let n be a natural number. Suppose $\text{ProgramPart}(\text{Relocated}(I, n)) \subseteq s_2$ and $\mathbf{IC}_{(s_2)} = \text{insloc}(n)$ and $s_1 \upharpoonright (\text{Int-Locations} \cup \text{FinSeq-Locations}) = s_2 \upharpoonright (\text{Int-Locations} \cup \text{FinSeq-Locations})$. Let i be a natural number. Then $\mathbf{IC}_{(\text{Computation}(s_1))(i)} + n = \mathbf{IC}_{(\text{Computation}(s_2))(i)}$ and $\text{IncAddr}(\text{CurInstr}((\text{Computation}(s_1))(i)), n) = \text{CurInstr}((\text{Computation}(s_2))(i))$ and $(\text{Computation}(s_1))(i) \upharpoonright (\text{Int-Locations} \cup \text{FinSeq-Locations}) = (\text{Computation}(s_2))(i) \upharpoonright (\text{Int-Locations} \cup \text{FinSeq-Locations})$.

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

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

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

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

The functor $if > 0(a, I, J)$ yields a macro instruction and is defined by:

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

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

(Def. 3) $if < 0(a, I, J) = if = 0(a, J, if > 0(a, J, I))$.

The following propositions are true:

(14) For all macro instructions I, J and for every integer location a holds $\text{card}\, if = 0(a, I, J) = \text{card}\, I + \text{card}\, J + 4$.

(15) For all macro instructions I, J and for every integer location a holds $\text{card}\, if > 0(a, I, J) = \text{card}\, I + \text{card}\, J + 4$.

(16) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) = 0$ and I is closed on

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

(17) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let 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($if = 0(a, I, J), s$) = IExec(I, s)+· Start-At(insloc(card I + card J + 3)).

(18) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) \neq 0$ and J is closed on s and halting on s. Then $if = 0(a, I, J)$ is closed on s and $if = 0(a, I, J)$ is halting on s.

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

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

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

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

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

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

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

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

(24) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) \leq 0$ and J is closed on s and halting on s. Then $if > 0(a, I, J)$ is closed on s and $if > 0(a, I, J)$ is halting on s.

(25) Let I, J be macro instructions, and let a be a read-write integer location, and let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $s(a) \leq 0$ and J is closed on

Initialize(s) and halting on Initialize(s). Then IExec($if > 0(a, I, J), s$) = IExec(J, s)+· Start-At(insloc(card I + card J + 3)).

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

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

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

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

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

(28) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) < 0$ and I is closed on s and halting on s. Then $if < 0(a, I, J)$ is closed on s and $if < 0(a, I, J)$ is halting on s.

(29) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let 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($if < 0(a, I, J), s$) = IExec(I, s)+· Start-At(insloc(card I + card J + card J + 7)).

(30) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) = 0$ and J is closed on s and halting on s. Then $if < 0(a, I, J)$ is closed on s and $if < 0(a, I, J)$ is halting on s.

(31) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let 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($if < 0(a, I, J), s$) = IExec(J, s)+· Start-At(insloc(card I + card J + card J + 7)).

(32) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let a be a read-write integer location. Suppose $s(a) > 0$ and J is closed on s and halting on s. Then $if < 0(a, I, J)$ is closed on s and $if < 0(a, I, J)$ is halting on s.

(33) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be macro instructions, and let 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($if < 0(a, I, J), s$) = IExec(J, s)+· Start-At(insloc(card I + card J + card J + 7)).

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

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

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

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

Let I, J be parahalting macro instructions and let a be a read-write integer location. Observe that $if = 0(a, I, J)$ is parahalting and $if > 0(a, I, J)$ is parahalting.

Let a, b be integer locations and let I, J be macro instructions. The functor $if = 0(a, b, I, J)$ yields a macro instruction and is defined as follows:

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

The functor $if > 0(a, b, I, J)$ yields a macro instruction and is defined by:

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

We introduce $if < 0(b, a, I, J)$ as a synonym of $if > 0(a, b, I, J)$.

Let I, J be parahalting macro instructions and let a, b be read-write integer locations. One can check that $if = 0(a, b, I, J)$ is parahalting and $if > 0(a, b, I, J)$ is parahalting.

Next we state several propositions:

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

(36) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I be a macro instruction, and let 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 $\mathbf{SCM}_{\mathrm{FSA}}$, and let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, and let 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 $\mathbf{SCM}_{\mathrm{FSA}}$, and let I be a macro instruction, and let 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 (Computation(s_1+$\cdot$(I+$\cdot$ Start-At(insloc(0)))))(k)(b) = (Computation(s_2+$\cdot$(I+$\cdot$ Start-At (insloc(0)))))(k)(b),

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

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

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

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

(40) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$, and let I be a macro instruction, and let 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 $\mathbf{SCM}_{\mathrm{FSA}}$, and let I be a macro instruction, and let 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 Initialize(s_1) and halting on Initialize(s_1).

Then

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

(vi) for every finite sequence location f holds (IExec(I, s_1))(f) = (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 $\mathbf{SCM}_{\mathrm{FSA}}$, and let I, J be parahalting macro instructions, and let a, b be read-write integer locations. Suppose I does not refer a and J does not refer a. Then

(i) $\mathbf{IC}_{\text{IExec}(if=0(a,b,I,J),s)} = \text{insloc}(\text{card}\, I + \text{card}\, J + 5)$,

(ii) if $s(a) = s(b)$, then for every integer location d such that $a \neq d$ holds (IExec($if = 0(a, b, I, J), s$))(d) = (IExec(I, s))(d) and for every finite sequence location f holds (IExec($if = 0(a, b, I, J), s$))(f) = (IExec(I, s))(f), and

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

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

(i) $\mathbf{IC}_{\mathrm{IExec}(if>0(a,b,I,J),s)} = \mathrm{insloc}(\mathrm{card}\, I + \mathrm{card}\, J + 5)$,

(ii) if $s(a) > s(b)$, then for every integer location d such that $a \neq d$ holds $(\mathrm{IExec}(if > 0(a,b,I,J),s))(d) = (\mathrm{IExec}(I,s))(d)$ and for every finite sequence location f holds $(\mathrm{IExec}(if > 0(a,b,I,J),s))(f) = (\mathrm{IExec}(I,s))(f)$, and

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

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

[3] Noriko Asamoto. Some multi instructions defined by sequence of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):615–619, 1996.

[4] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Formalized Mathematics*, 6(**1**):53–57, 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] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.

[7] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.

[8] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.

[9] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **SCM**. *Formalized Mathematics*, 4(**1**):61–67, 1993.

[10] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.

[11] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.

[12] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.

[13] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.

[14] 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.

[15] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.

[16] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.

[17] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.

[18] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.

[19] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[20] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[21] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[22] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[23] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[24] Andrzej Trybulec and Yatsuka Nakamura. Relocability for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):583–586, 1996.
[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, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
[27] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **SCM**. *Formalized Mathematics*, 5(**4**):507–512, 1996.
[28] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[29] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[30] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[31] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[32] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received August 27, 1996