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

The `while` Macro Instructions of $\mathbf{SCM_{FSA}}$. Part II

Piotr Rudnicki
University of Alberta
Edmonton

Summary. An attempt to use the `while` macro, [16], was the origin of writing this article. The `while` semantics, as given by J.-C. Chen, is slightly extended by weakening its correctness conditions and this forced a quite straightforward remake of a number of theorems from [16]. Numerous additional properties of the `while` macro are then proven. In the last section, we define a macro instruction computing the `fusc` function (see the SCM program computing the same function in [12]) and prove its correctness.

MML Identifier: SCMFSA9A.

WWW: http://mizar.org/JFM/Vol10/scmfsa9a.html

The articles [23], [32], [24], [25], [22], [7], [9], [30], [8], [20], [33], [13], [14], [15], [11], [17], [26], [10], [21], [29], [27], [28], [5], [19], [6], [4], [3], [1], [2], [16], [18], and [31] provide the notation and terminology for this paper.

1. Arithmetic Preliminaries

We adopt the following convention: k, m, n are natural numbers, i, j are integers, and r is a real number.

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

There exists k such that $\mathcal{P}[k]$ and for every n such that $\mathcal{P}[n]$ holds $k \leq n$

provided the parameters meet the following condition:

- For every k holds $\mathcal{F}(k+1) < \mathcal{F}(k)$ or $\mathcal{P}[k]$.

One can prove the following propositions:

(1) n is odd iff there exists a natural number k such that $n = 2 \cdot k + 1$.

(2) For every integer i such that $i \leq r$ holds $i \leq \lfloor r \rfloor$.

(3) If $0 < n$, then $0 \leq (m$ **qua** integer$) \div n$.

(4) If $0 < i$ and $1 < j$, then $i \div j < i$.

(5) $(m$ **qua** integer$) \div n = m \div n$ and $(m$ **qua** integer$) \bmod n = m \bmod n$.

2. $\mathbf{SCM_{FSA}}$ Preliminaries

In the sequel l denotes an instruction-location of $\mathbf{SCM_{FSA}}$ and i denotes an instruction of $\mathbf{SCM_{FSA}}$.

We now state several propositions:

(6) Let N be a non empty set with non empty elements, S be a halting IC-Ins-separated definite non empty non void AMI over N, s be a state of S, and k be a natural number. If $\mathrm{CurInstr}((\mathrm{Computation}(s))(k)) = \mathbf{halt}_S$, then $(\mathrm{Computation}(s))(\mathrm{LifeSpan}(s)) = (\mathrm{Computation}(s))(k)$.

(7) $\mathrm{UsedIntLoc}(l \dot{\longmapsto} i) = \mathrm{UsedIntLoc}(i)$.

(8) $\mathrm{UsedInt^*\,Loc}(l \dot{\longmapsto} i) = \mathrm{UsedInt^*\,Loc}(i)$.

(9) $\mathrm{UsedIntLoc}(\mathrm{Stop}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \emptyset$.

(10) $\mathrm{UsedInt^*\,Loc}(\mathrm{Stop}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \emptyset$.

(11) $\mathrm{UsedIntLoc}(\mathrm{Goto}(l)) = \emptyset$.

(12) $\mathrm{UsedInt^*\,Loc}(\mathrm{Goto}(l)) = \emptyset$.

For simplicity, we follow the rules: s, s_1, s_2 denote states of $\mathbf{SCM}_{\mathrm{FSA}}$, a denotes a read-write integer location, b denotes an integer location, I, J denote macro instructions, I_1 denotes a good macro instruction, and i, j, k denote natural numbers.

One can prove the following four propositions:

(13) $\mathrm{UsedIntLoc}(\mathbf{if}\ b = 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \{b\} \cup \mathrm{UsedIntLoc}(I) \cup \mathrm{UsedIntLoc}(J)$.

(14) For every integer location a holds $\mathrm{UsedInt^*\,Loc}(\mathbf{if}\ a = 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \mathrm{UsedInt^*\,Loc}(I) \cup \mathrm{UsedInt^*\,Loc}(J)$.

(15) $\mathrm{UsedIntLoc}(\mathbf{if}\ b > 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \{b\} \cup \mathrm{UsedIntLoc}(I) \cup \mathrm{UsedIntLoc}(J)$.

(16) $\mathrm{UsedInt^*\,Loc}(\mathbf{if}\ b > 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \mathrm{UsedInt^*\,Loc}(I) \cup \mathrm{UsedInt^*\,Loc}(J)$.

3. THE while=0 MACRO INSTRUCTION

The following two propositions are true:

(17) $\mathrm{UsedIntLoc}(\mathbf{while}\ b = 0\ \mathbf{do}\ I) = \{b\} \cup \mathrm{UsedIntLoc}(I)$.

(18) $\mathrm{UsedInt^*\,Loc}(\mathbf{while}\ b = 0\ \mathbf{do}\ I) = \mathrm{UsedInt^*\,Loc}(I)$.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, let a be a read-write integer location, and let I be a macro instruction. The predicate $\mathrm{ProperBodyWhile{=}0}(a,I,s)$ is defined as follows:

(Def. 1) For every natural number k such that $(\mathit{StepWhile{=}0}(a,I,s))(k)(a) = 0$ holds I is closed on $(\mathit{StepWhile{=}0}(a,I,s))(k)$ and halting on $(\mathit{StepWhile{=}0}(a,I,s))(k)$.

The predicate $\mathrm{WithVariantWhile{=}0}(a,I,s)$ is defined by the condition (Def. 2).

(Def. 2) 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 $(\mathit{StepWhile{=}0}(a,I,s))(k)(a) \neq 0$.

We now state several propositions:

(19) For every parahalting macro instruction I holds $\mathrm{ProperBodyWhile{=}0}(a,I,s)$.

(20) If $\mathrm{ProperBodyWhile{=}0}(a,I,s)$ and $\mathrm{WithVariantWhile{=}0}(a,I,s)$, then $\mathbf{while}\ a = 0\ \mathbf{do}\ I$ is halting on s and $\mathbf{while}\ a = 0\ \mathbf{do}\ I$ is closed on s.

(21) For every parahalting macro instruction I such that $\mathrm{WithVariantWhile{=}0}(a,I,s)$ holds $\mathbf{while}\ a = 0\ \mathbf{do}\ I$ is halting on s and $\mathbf{while}\ a = 0\ \mathbf{do}\ I$ is closed on s.

(22) If $(\mathbf{while}\ a = 0\ \mathbf{do}\ I) + \cdot S_1 \subseteq s$ and $s(a) \neq 0$, then $\mathrm{LifeSpan}(s) = 4$ and for every natural number k holds $(\mathrm{Computation}(s))(k) \restriction D = s \restriction D$, where $S_1 = \text{Start-At}(\mathrm{insloc}(0))$ and $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(23) If I is closed on s and halting on s and $s(a) = 0$, then $(\mathrm{Computation}(s + \cdot ((\mathbf{while}\ a = 0\ \mathbf{do}\ I) + \cdot \text{Start-At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(s + \cdot (I + \cdot \text{Start-At}(\mathrm{insloc}(0)))) + 3) \restriction D = (\mathrm{Computation}(s + \cdot (I + \cdot \text{Start-At}(\mathrm{insloc}($ where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(24) If $(\mathit{StepWhile{=}0}(a, I, s))(k)(a) \neq 0$, then $(\mathit{StepWhile{=}0}(a, I, s))(k+1) \restriction D = (\mathit{StepWhile{=}0}(a, I, s))(k) \restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(25) Suppose I is halting on $\mathrm{Initialize}((\mathit{StepWhile{=}0}(a, I, s))(k))$, closed on $\mathrm{Initialize}((\mathit{StepWhile{=}0}(a, I, s))(k))$, and parahalting and $(\mathit{StepWhile{=}0}(a, I, s))(k)(a) = 0$ and $(\mathit{StepWhile{=}0}(a, I, s))(k)(\mathrm{intloc}(0)) = 1$. Then $(\mathit{StepWhile{=}0}(a, I, s))(k+1) \restriction D = \mathrm{IExec}(I, (\mathit{StepWhile{=}0}(a, I, s))(k)) \restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(26) If $\mathrm{ProperBodyWhile{=}0}(a, I_1, s)$ or I_1 is parahalting and if $s(\mathrm{intloc}(0)) = 1$, then for every k holds $(\mathit{StepWhile{=}0}(a, I_1, s))(k)(\mathrm{intloc}(0)) = 1$.

(27) If $\mathrm{ProperBodyWhile{=}0}(a, I, s_1)$ and $s_1 \restriction D = s_2 \restriction D$, then for every k holds $(\mathit{StepWhile{=}0}(a, I, s_1))(k) \restriction D = (\mathit{StepWhile{=}0}(a, I, s_2))(k) \restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, let a be a read-write integer location, and let I be a macro instruction. Let us assume that $\mathrm{ProperBodyWhile{=}0}(a, I, s)$ or I is parahalting and $\mathrm{WithVariantWhile{=}0}(a, I, s)$. The functor $\mathit{ExitsAtWhile{=}0}(a, I, s)$ yields a natural number and is defined by the condition (Def. 3).

(Def. 3) There exists a natural number k such that

(i) $\mathit{ExitsAtWhile{=}0}(a, I, s) = k$,

(ii) $(\mathit{StepWhile{=}0}(a, I, s))(k)(a) \neq 0$,

(iii) for every natural number i such that $(\mathit{StepWhile{=}0}(a, I, s))(i)(a) \neq 0$ holds $k \leq i$, and

(iv) $(\mathrm{Computation}(s + \cdot ((\mathbf{while}\ a = 0\ \mathbf{do}\ I) + \cdot S_1)))(\mathrm{LifeSpan}(s + \cdot ((\mathbf{while}\ a = 0\ \mathbf{do}\ I) + \cdot S_1))) \restriction D = (\mathit{StepWhile{=}0}(a, I, s))(k) \restriction D$,

where $S_1 = \text{Start-At}(\mathrm{insloc}(0))$ and $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

Next we state two propositions:

(28) If $s(\mathrm{intloc}(0)) = 1$ and $s(a) \neq 0$, then $\mathrm{IExec}(\mathbf{while}\ a = 0\ \mathbf{do}\ I, s) \restriction D = s \restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(29) If $\mathrm{ProperBodyWhile{=}0}(a, I, \mathrm{Initialize}(s))$ or I is parahalting and if $\mathrm{WithVariantWhile{=}0}(a, I, \mathrm{Initialize}(s))$, then $\mathrm{IExec}(\mathbf{while}\ a = 0\ \mathbf{do}\ I, s) \restriction D = (\mathit{StepWhile{=}0}(a, I, \mathrm{Initialize}(s)))(\mathit{ExitsAtWhile{=}0}(a, I, \mathrm{Initialize}(s))) \restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

4. THE while>0 MACRO INSTRUCTION

Next we state two propositions:

(30) $\mathrm{UsedIntLoc}(\mathbf{while}\ b > 0\ \mathbf{do}\ I) = \{b\} \cup \mathrm{UsedIntLoc}(I)$.

(31) $\mathrm{UsedInt}^{*}\mathrm{Loc}(\mathbf{while}\ b > 0\ \mathbf{do}\ I) = \mathrm{UsedInt}^{*}\mathrm{Loc}(I)$.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, let a be a read-write integer location, and let I be a macro instruction. The predicate $\mathrm{ProperBodyWhile{>}0}(a, I, s)$ is defined as follows:

(Def. 4) For every natural number k such that $(\mathit{StepWhile{>}0}(a, I, s))(k)(a) > 0$ holds I is closed on $(\mathit{StepWhile{>}0}(a, I, s))(k)$ and halting on $(\mathit{StepWhile{>}0}(a, I, s))(k)$.

The predicate $\mathrm{WithVariantWhile{>}0}(a, I, s)$ is defined by the condition (Def. 5).

(Def. 5) 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 $(\mathit{StepWhile}{>}0(a,I,s))(k)(a) \leq 0$.

We now state several propositions:

(32) For every parahalting macro instruction I holds ProperBodyWhile$>0(a,I,s)$.

(33) If ProperBodyWhile$>0(a,I,s)$ and WithVariantWhile$>0(a,I,s)$, then **while** $a>0$ **do** I is halting on s and **while** $a>0$ **do** I is closed on s.

(34) For every parahalting macro instruction I such that WithVariantWhile$>0(a,I,s)$ holds **while** $a>0$ **do** I is halting on s and **while** $a>0$ **do** I is closed on s.

(35) If $($**while** $a>0$ **do** $I)+\cdot S_1 \subseteq s$ and $s(a) \leq 0$, then LifeSpan$(s)=4$ and for every natural number k holds $(\mathrm{Computation}(s))(k)\restriction D = s\restriction D$, where $S_1 = \mathrm{Start\text{-}At}(\mathrm{insloc}(0))$ and $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(36) If I is closed on s and halting on s and $s(a)>0$, then (Computation($s+\cdot(($**while** $a>0$ **do** $I)+\cdot$ Start-At(insloc(0)))))(LifeSpan($s+\cdot(I+\cdot$ Start-At(insloc(0))))$+3)\restriction D =$ (Computation($s+\cdot(I+\cdot$ Start-At(insloc(where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(37) If $(\mathit{StepWhile}{>}0(a,I,s))(k)(a) \leq 0$, then $(\mathit{StepWhile}{>}0(a,I,s))(k+1)\restriction D = (\mathit{StepWhile}{>}0(a,I,s))(k)\restriction D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(38) Suppose I is halting on Initialize$((\mathit{StepWhile}{>}0(a,I,s))(k))$, closed on Initialize$((\mathit{StepWhile}{>}0(a,I,s))(k))$, and parahalting and $(\mathit{StepWhile}{>}0(a,I,s))(k)(a)>0$ and $(\mathit{StepWhile}{>}0(a,I,s))(k)(\mathrm{intloc}(0)) = 1$. Then $(\mathit{StepWhile}{>}0(a,I,s))(k+1)\restriction D = \mathrm{IExec}(I,(\mathit{StepWhile}{>}0(a,I,s))(k))\restriction D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(39) If ProperBodyWhile$>0(a,I_1,s)$ or I_1 is parahalting and if $s(\mathrm{intloc}(0))=1$, then for every k holds $(\mathit{StepWhile}{>}0(a,I_1,s))(k)(\mathrm{intloc}(0))=1$.

(40) If ProperBodyWhile$>0(a,I,s_1)$ and $s_1\restriction D = s_2\restriction D$, then for every k holds $(\mathit{StepWhile}{>}0(a,I,s_1))(k)\restriction D = (\mathit{StepWhile}{>}0(a,I,s_2))(k)\restriction D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, let a be a read-write integer location, and let I be a macro instruction. Let us assume that ProperBodyWhile$>0(a,I,s)$ or I is parahalting and WithVariantWhile$>0(a,I,s)$. The functor $\mathit{ExitsAtWhile}{>}0(a,I,s)$ yielding a natural number is defined by the condition (Def. 6).

(Def. 6) There exists a natural number k such that

(i) $\mathit{ExitsAtWhile}{>}0(a,I,s)=k$,

(ii) $(\mathit{StepWhile}{>}0(a,I,s))(k)(a) \leq 0$,

(iii) for every natural number i such that $(\mathit{StepWhile}{>}0(a,I,s))(i)(a) \leq 0$ holds $k \leq i$, and

(iv) (Computation($s+\cdot(($**while** $a>0$ **do** $I)+\cdot S_1)$))(LifeSpan($s+\cdot(($**while** $a>0$ **do** $I)+\cdot S_1)$))$\restriction D = (\mathit{StepWhile}{>}0(a,I,s))(k)\restriction D$,

where $S_1 = \mathrm{Start\text{-}At}(\mathrm{insloc}(0))$ and $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

Next we state several propositions:

(41) If $s(\mathrm{intloc}(0))=1$ and $s(a)\leq 0$, then IExec(**while** $a>0$ **do** $I,s)\restriction D = s\restriction D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(42) If ProperBodyWhile$>0(a,I,\mathrm{Initialize}(s))$ or I is parahalting and if WithVariantWhile$>0(a,I,\mathrm{Initialize}(s))$, then IExec(**while** $a>0$ **do** $I,s)\restriction D = (\mathit{StepWhile}{>}0(a,I,\mathrm{Initialize}(s)))(\mathit{ExitsAtWhile}{>}0(a,I,\mathrm{Initialize}(s)))\restriction D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

(43) If $(\mathit{StepWhile}{>}\mathit{0}(a,I,s))(k)(a) \leq 0$, then for every natural number n such that $k \leq n$ holds $(\mathit{StepWhile}{>}\mathit{0}(a,I,s))(n){\upharpoonright}D = (\mathit{StepWhile}{>}\mathit{0}(a,I,s))(k){\upharpoonright}D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(44) If $s_1{\upharpoonright}D = s_2{\upharpoonright}D$ and $\text{ProperBodyWhile}{>}0(a,I,s_1)$, then $\text{ProperBodyWhile}{>}0(a,I,s_2)$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(45) Suppose $s(\text{intloc}(0)) = 1$ and $\text{ProperBodyWhile}{>}0(a,I_1,s)$ and $\text{WithVariantWhile}{>}0(a,I_1,s)$. Let given i, j. Suppose $i \neq j$ and $i \leq \mathit{ExitsAtWhile}{>}\mathit{0}(a,I_1,s)$ and $j \leq \mathit{ExitsAtWhile}{>}\mathit{0}(a,I_1,s)$. Then $(\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(i) \neq (\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(j)$ and $(\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(i){\upharpoonright}D \neq (\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(j){\upharpoonright}D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

Let f be a function from $\prod$(the object kind of $\mathbf{SCM}_{\text{FSA}}$) into $\mathbb{N}$. We say that f is on data only if and only if:

(Def. 7) For all s_1, s_2 such that $s_1{\upharpoonright}D = s_2{\upharpoonright}D$ holds $f(s_1) = f(s_2)$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

We now state two propositions:

(46) Suppose $s(\text{intloc}(0)) = 1$ and $\text{ProperBodyWhile}{>}0(a,I_1,s)$ and $\text{WithVariantWhile}{>}0(a,I_1,s)$. Then there exists a function f from $\prod$(the object kind of $\mathbf{SCM}_{\text{FSA}}$) into $\mathbb{N}$ such that f is on data only and for every natural number k holds $f((\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(k+1)) < f((\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(k))$ or $(\mathit{StepWhile}{>}\mathit{0}(a,I_1,s))(k)(a) \leq 0$.

(47) If $s_1(\text{intloc}(0)) = 1$ and $s_1{\upharpoonright}D = s_2{\upharpoonright}D$ and $\text{ProperBodyWhile}{>}0(a,I_1,s_1)$ and $\text{WithVariantWhile}{>}0(a,I_1,s_1)$, then $\text{WithVariantWhile}{>}0(a,I_1,s_2)$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

5. A MACRO FOR THE fusc FUNCTION

Let N, r_1 be integer locations. The functor $\text{Fusc\_macro}(N,r_1)$ yielding a macro instruction is defined by:

(Def. 8) $\text{Fusc\_macro}(N,r_1) = \text{SubFrom}(r_1,r_1);\ (n_1{:=}\text{intloc}(0));\ (a_1{:=}N);\ (\mathbf{while}\ a_1 > 0\ \mathbf{do}\ ((r_2{:=}2);\ \text{Divide}(a_1,r_2);\ (\mathbf{if}\ r_2 = 0\ \mathbf{then}\ \text{Macro}(\text{AddTo}(n_1,r_1))\ \mathbf{else}\ \text{Macro}(\text{AddTo}(r_1,n_1)))))$, where $n_1 = 1^{\text{st}}\text{-RWNotIn}(\{N,r_1\})$, $a_1 = 2^{\text{nd}}\text{-RWNotIn}(\{N,r_1\})$, and $r_2 = 3^{\text{rd}}\text{-RWNotIn}(\{N,r_1\})$.

One can prove the following proposition

(48) Let N, r_1 be read-write integer locations. Suppose $N \neq r_1$. Let n be a natural number. If $n = s(N)$, then $(\text{IExec}(\text{Fusc\_macro}(N,r_1),s))(r_1) = \text{Fusc}(n)$ and $(\text{IExec}(\text{Fusc\_macro}(N,r_1),s))(N) = n$.

REFERENCES

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

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

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

[4] Noriko Asamoto. Some multi instructions defined by sequence of instructions of $\mathbf{SCM}_{\text{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa_7.html.

[5] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa6b.html.

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

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

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

[9] Grzegorz Bancerek. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/card_3.html`.

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

[11] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part I - preliminaries. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/pre_ff.html`.

[12] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part II - programs. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/fib_fusc.html`.

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

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

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

[16] Jing-Chao Chen. While macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 9, 1997. `http://mizar.org/JFM/Vol9/scmfsa_9.html`.

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

[18] Piotr Rudnicki. On the composition of non-parahalting macro instructions. *Journal of Formalized Mathematics*, 10, 1998. `http://mizar.org/JFM/Vol10/sfmastr1.html`.

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

[20] Piotr Rudnicki and Andrzej Trybulec. Abian's fixed point theorem. *Journal of Formalized Mathematics*, 9, 1997. `http://mizar.org/JFM/Vol9/abian.html`.

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

[22] Andrzej Trybulec. Binary operations applied to functions. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funcop_1.html`.

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

[24] Andrzej Trybulec. Subsets of real numbers. *Journal of Formalized Mathematics*, Addenda, 2003. `http://mizar.org/JFM/Addenda/numbers.html`.

[25] Andrzej Trybulec and Agata Darmochwał. Boolean domains. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/finsub_1.html`.

[26] 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`.

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

[28] 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`.

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

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

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

[32] Zinaida Trybulec. Properties of subsets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/subset_1.html`.

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

Received June 3, 1998

Published January 2, 2004