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

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

Piotr Rudnicki[1]
University of Alberta
Edmonton

**Summary.** An attempt to use the `while` macro, [14], 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 [14]. 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 [10]) and prove its correctness.

MML Identifier: `SCMFSA9A`.

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

## 1. Arithmetic Preliminaries

We follow the rules: $k$, $m$, $n$ 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 $k \leqslant n$

provided the parameters meet the following condition:

[1]This work was partially supported by NSERC Grant OGP9207 and NATO CRG 951368.




ISSN 1426–2630

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

We now state several propositions:

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

(2) If $0 \leqslant r$, then $0 \leqslant \lfloor r \rfloor$.

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

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

(5) If $0 < n$, then $(m$ **qua** integer$) \div n = m \div n$ and $(m$ **qua** integer$) \bmod n = m \bmod n$.

## 2. $\mathbf{SCM}_{\mathrm{FSA}}$ Preliminaries

In the sequel $l$ is an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$ and $i$ is an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$.

Next we state several propositions:

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

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

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

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

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

(11) UsedIntLoc(Goto($l$)) $= \emptyset$.

(12) UsedInt$^*$ Loc(Goto($l$)) $= \emptyset$.

For simplicity, we use the following convention: $s$, $s_1$, $s_2$ are states of $\mathbf{SCM}_{\mathrm{FSA}}$, $a$ is a read-write integer location, $b$ is an integer location, $f$ is a finite sequence location, $I$, $J$ are macro instructions, $I_1$ is a good macro instruction, and $i$, $j$, $k$ are natural numbers.

The following four propositions are true:

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

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

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

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

## 3. The while=0 Macro Instruction

Next we state two propositions:

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

(18) UsedInt$^*$ Loc(**while** $b = 0$ **do** $I$) = UsedInt$^*$ Loc($I$).

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

(20) 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 $s$.

(21) 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 $s$.

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

(23) If $I$ is closed on $s$ and halting on $s$ and $s(a) = 0$, then (Computation($s+\cdot$((**while** $a = 0$ **do** $I$)$+\cdot S_1$)))(LifeSpan($s+\cdot(I+\cdot S_1)$) + 3)$\restriction D$ = (Computation($s+\cdot(I+\cdot S_1)$))(LifeSpan($s+\cdot(I+\cdot S_1)$))$\restriction D$, where $S_1$ = Start-At(insloc(0)) and $D$ = Int-Locations ∪ 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 $D$ = Int-Locations ∪ FinSeq-Locations.

(25) 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 $(\mathit{StepWhile{=}0}(a, I, s))(k{+}1)\restriction D$ = IExec($I$, $(\mathit{StepWhile{=}0}(a, I, s))(k)$)$\restriction D$, where $D$ = Int-Locations ∪ FinSeq-Locations.

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

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

Let $s$ be a state of $\mathbf{SCM}_{\text{FSA}}$, 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 $\textit{ExitsAtWhile=0}(a, I, s)$ yielding a natural number is defined by the condition (Def. 3).

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

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

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

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

(iv) $(\text{Computation}(s +\cdot ((\textbf{while } a = 0 \textbf{ do } I) +\cdot S_1)))(\text{LifeSpan}(s +\cdot ((\textbf{while } a = 0 \textbf{ do } I) +\cdot S_1))) \upharpoonright D = (\textit{StepWhile=0}(a, I, s))(k) \upharpoonright D$,

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

One can prove the following two propositions:

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

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

## 4. The `while>0` Macro Instruction

The following propositions are true:

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

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

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

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

The predicate 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 $\mathbf{SCM}_{\rm FSA}$) 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 $(\mathit{StepWhile{>}0}(a,I,s))(k)(a) \leqslant 0$.

Next we 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 $s(a) \leqslant 0$, then LifeSpan$(s)=4$ and for every natural number $k$ holds (Computation$(s))(k){\restriction}D = s{\restriction}D$, where $S_1 =$ Start-At(insloc(0)) and $D =$ Int-Locations $\cup$ FinSeq-Locations.

(36) If $I$ is closed on $s$ and halting on $s$ and $s(a)>0$, then (Computation($s+\cdot$((**while** $a>0$ **do** $I$)$+\cdot S_1$)))(LifeSpan($s+\cdot(I+\cdot S_1)) + 3){\restriction}D$ = (Computation($s+\cdot(I+\cdot S_1)$))(LifeSpan($s+\cdot(I+\cdot S_1)$))${\restriction}D$, where $S_1 =$ Start-At(insloc(0)) and $D =$ Int-Locations $\cup$ FinSeq-Locations.

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

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

(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 $D =$ Int-Locations $\cup$ FinSeq-Locations.

Let $s$ be a state of $\mathbf{SCM}_{\rm FSA}$, 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 $\mathit{ExitsAtWhile{>}0}(a,I,s)$ yields a natural number and 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) $(\mathit{StepWhile{>}0}(a,I,s))(k)(a) \leqslant 0$,

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

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

where $S_1$ = Start-At(insloc(0)) and $D$ = Int-Locations ∪ FinSeq-Locations.

Next we state several propositions:

(41) If $s$(intloc(0)) = 1 and $s(a) \leqslant 0$, then IExec(**while** $a > 0$ **do** $I, s$)$\restriction D = s\restriction D$, where $D$ = Int-Locations ∪ FinSeq-Locations.

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

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

(44) If $s_1\restriction D = s_2\restriction D$ and ProperBodyWhile>0($a, I, s_1$), then

ProperBodyWhile>0($a, I, s_2$), where $D$ = Int-Locations ∪ FinSeq-Locations.

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

Let $f$ be a function from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) 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 $s_1\restriction D = s_2\restriction D$ holds $f(s_1) = f(s_2)$, where $D$ = Int-Locations ∪ FinSeq-Locations.

We now state two propositions:

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

(47) If $s_1$(intloc(0)) = 1 and $s_1\restriction D = s_2\restriction D$ and ProperBodyWhile>0($a, I_1, s_1$) and WithVariantWhile>0($a, I_1, s_1$), then WithVariantWhile>0($a, I_1, s_2$), where $D$ = Int-Locations ∪ FinSeq-Locations.

## 5. A Macro for the fusc Function

Let $N$, $r_1$ be integer locations. The functor Fusc_macro$(N, r_1)$ yields a macro instruction and is defined as follows:

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

## References


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

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

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

[4] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Formalized Mathematics*, 6(**1**):41–47, 1997.

[5] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Formalized Mathematics*, 6(**1**):53–57, 1997.

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

[7] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.

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

[9] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part I - preliminaries. *Formalized Mathematics*, 4(**1**):69–72, 1993.

[10] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part II - programs. *Formalized Mathematics*, 4(**1**):73–75, 1993.

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

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

[13] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.

[14] Jing-Chao Chen. While macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**4**):553–561, 1997.
[15] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[16] Piotr Rudnicki. On the composition of non-parahalting macro instructions. *Formalized Mathematics*, 7(**1**):87–92, 1998.
[17] Piotr Rudnicki and Andrzej Trybulec. Abian's fixed point theorem. *Formalized Mathematics*, 6(**3**):335–338, 1997.
[18] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[19] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[20] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[21] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[22] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[23] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[24] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[25] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[26] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[27] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[28] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.
[29] Wojciech Zielonka. Preliminaries to the Lambek calculus. *Formalized Mathematics*, 2(**3**):413–418, 1991.

*Received June 3, 1998*