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

The `for` (going up) Macro Instruction

Piotr Rudnicki
University of Alberta
Edmonton

Summary. We define a `for` type (going up) macro instruction in terms of the `while` macro. This gives an iterative macro with an explicit control variable. The `for` macro is used to define a macro for the selection sort acting on a finite sequence location of $\mathbf{SCM}_{\rm FSA}$. On the way, a macro for finding a minimum in a section of an array is defined.

MML Identifier: SFMASTR3.

WWW: http://mizar.org/JFM/Vol10/sfmastr3.html

The articles [24], [35], [7], [26], [9], [8], [18], [25], [32], [6], [33], [36], [37], [12], [14], [13], [11], [19], [5], [17], [27], [23], [10], [15], [34], [20], [28], [31], [29], [30], [3], [22], [4], [2], [1], [16], and [21] provide the notation and terminology for this paper.

1. General Preliminaries

One can prove the following two propositions:

(1) Let X be a set, p be a permutation of X, and x, y be elements of X. Then $p +\cdot (x, p(y)) +\cdot (y, p(x))$ is a permutation of X.

(2) Let f be a function and x, y be sets. Suppose $x \in \operatorname{dom} f$ and $y \in \operatorname{dom} f$. Then there exists a permutation p of $\operatorname{dom} f$ such that $f +\cdot (x, f(y)) +\cdot (y, f(x)) = f \cdot p$.

Let A be a finite non empty real-membered set. Then $\inf A$ can be characterized by the condition:

(Def. 1) $\inf A \in A$ and for every real number k such that $k \in A$ holds $\inf A \leq k$.

We introduce $\min A$ as a synonym of $\inf A$.

Let X be a finite non empty natural-membered set. Observe that $\min X$ is integer.

Let F be a finite sequence of elements of $\mathbb{Z}$ and let m, n be natural numbers. Let us assume that $1 \leq m$ and $m \leq n$ and $n \leq \operatorname{len} F$. The functor $\min_m^n F$ yields a natural number and is defined as follows:

(Def. 3)[1] There exists a finite non empty subset X of $\mathbb{Z}$ such that $X = \operatorname{rng}\langle F(m), \ldots, F(n)\rangle$ and $(\min_m^n F) + 1 = (\min X) \leftarrow \langle F(m), \ldots, F(n)\rangle + m$.

We follow the rules: F, F_1 are finite sequences of elements of $\mathbb{Z}$ and k, m, n, m_1 are natural numbers.

We now state two propositions:

[1] The definition (Def. 2) has been removed.

(3) Suppose $1 \leq m$ and $m \leq n$ and $n \leq \operatorname{len} F$. Then $m_1 = \min_m^n F$ if and only if the following conditions are satisfied:

(i) $m \leq m_1$,

(ii) $m_1 \leq n$,

(iii) for every natural number i such that $m \leq i$ and $i \leq n$ holds $F(m_1) \leq F(i)$, and

(iv) for every natural number i such that $m \leq i$ and $i < m_1$ holds $F(m_1) < F(i)$.

(4) If $1 \leq m$ and $m \leq \operatorname{len} F$, then $\min_m^m F = m$.

Let F be a finite sequence of elements of $\mathbb{Z}$ and let m, n be natural numbers. We say that F is non decreasing on m, n if and only if:

(Def. 4) For all natural numbers i, j such that $m \leq i$ and $i \leq j$ and $j \leq n$ holds $F(i) \leq F(j)$.

Let F be a finite sequence of elements of $\mathbb{Z}$ and let n be a natural number. We say that F is split at n if and only if:

(Def. 5) For all natural numbers i, j such that $1 \leq i$ and $i \leq n$ and $n < j$ and $j \leq \operatorname{len} F$ holds $F(i) \leq F(j)$.

One can prove the following two propositions:

(5) Suppose $k+1 \leq \operatorname{len} F$ and $m_1 = \min_{(k+1)}^{(\operatorname{len} F)} F$ and F is split at k and F is non decreasing on 1, k and $F_1 = F +\cdot (k+1, F(m_1)) +\cdot (m_1, F(k+1))$. Then F_1 is non decreasing on 1, $k+1$.

(6) If $k+1 \leq \operatorname{len} F$ and $m_1 = \min_{(k+1)}^{(\operatorname{len} F)} F$ and F is split at k and $F_1 = F +\cdot (k+1, F(m_1)) +\cdot (m_1, F(k+1))$, then F_1 is split at $k+1$.

2. $\mathbf{SCM}_{\mathrm{FSA}}$ PRELIMINARIES

For simplicity, we adopt the following rules: s is a state of $\mathbf{SCM}_{\mathrm{FSA}}$, a, c are read-write integer locations, a_1, b_1, c_1, d_1, x are integer locations, f is a finite sequence location, I, J are macro instructions, I_1 is a good macro instruction, and k is a natural number.

The following propositions are true:

(7) If I is closed on Initialize(s) and halting on Initialize(s) and I does not destroy a_1, then $(\mathrm{IExec}(I,s))(a_1) = (\mathrm{Initialize}(s))(a_1)$.

(8) If $s(\mathrm{intloc}(0)) = 1$, then $\mathrm{IExec}(\mathrm{Stop}_{\mathbf{SCM}_{\mathrm{FSA}}}, s) \upharpoonright D = s \upharpoonright D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(9) $\mathrm{Stop}_{\mathbf{SCM}_{\mathrm{FSA}}}$ does not refer a_1.

(10) If $a_1 \neq b_1$, then $c_1 := b_1$ does not refer a_1.

(11) $(\mathrm{Exec}(a := f_{b_1}, s))(a) = s(f)_{|s(b_1)|}$.

(12) $(\mathrm{Exec}(f_{a_1} := b_1, s))(f) = s(f) +\cdot (|s(a_1)|, s(b_1))$.

Let a be a read-write integer location, let b be an integer location, and let I, J be good macro instructions. Observe that **if** $a > b$ **then** I **else** J is good.

The following propositions are true:

(13) UsedIntLoc(**if** $a_1 > b_1$ **then** I **else** J) $= \{a_1, b_1\} \cup \mathrm{UsedIntLoc}(I) \cup \mathrm{UsedIntLoc}(J)$.

(14) If I does not destroy a_1, then **while** $b_1 > 0$ **do** I does not destroy a_1.

(15) If $c_1 \neq a_1$ and I does not destroy c_1 and J does not destroy c_1, then **if** $a_1 > b_1$ **then** I **else** J does not destroy c_1.

3. THE for-up MACRO INSTRUCTION

Let a, b, c be integer locations, let I be a macro instruction, and let s be a state of $\mathbf{SCM}_{\rm FSA}$. The functor $\mathrm{StepForUp}(a,b,c,I,s)$ yields a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\rm FSA}$) and is defined as follows:

(Def. 6) $\mathrm{StepForUp}(a,b,c,I,s) = \mathit{StepWhile{>}0}(a_2,I;\ \mathrm{AddTo}(a,\mathrm{intloc}(0));\ \mathrm{SubFrom}(a_2,\mathrm{intloc}(0)), s+\cdot (a_2,(s(c)-s(b))+1)+\cdot(a,s(b)))$, where $a_2 = 1^{\rm st}\text{-RWNotIn}(\{a,b,c\}\cup\mathrm{UsedIntLoc}(I))$.

We now state several propositions:

(16) If $s(\mathrm{intloc}(0)) = 1$, then $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(\mathrm{intloc}(0)) = 1$.

(17) $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(a) = s(b_1)$.

(18) If $a \neq b_1$, then $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(b_1) = s(b_1)$.

(19) If $a \neq c_1$, then $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(c_1) = s(c_1)$.

(20) If $a \neq d_1$ and $d_1 \in \mathrm{UsedIntLoc}(I)$, then $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(d_1) = s(d_1)$.

(21) $(\mathrm{StepForUp}(a,b_1,c_1,I,s))(0)(f) = s(f)$.

(22) Suppose $s(\mathrm{intloc}(0)) = 1$. Let a_2 be a read-write integer location. If $a_2 = 1^{\rm st}\text{-RWNotIn}(\{a,b_1,c_1\}\cup\mathrm{UsedIntLoc}(I))$, then $\mathrm{IExec}((a_2{:=}c_1);\ \mathrm{SubFrom}(a_2,b_1);\ \mathrm{AddTo}(a_2,\mathrm{intloc}(0));\ (a{:=}b_1),s)\restriction$ $(s+\cdot(a_2,(s(c_1)-s(b_1))+1)+\cdot(a,s(b_1)))\restriction D$, where $D = \text{Int-Locations}\cup\text{FinSeq-Locations}$.

Let a, b, c be integer locations, let I be a macro instruction, and let s be a state of $\mathbf{SCM}_{\rm FSA}$. We say that ProperForUpBody a, b, c, I, s if and only if:

(Def. 7) For every natural number i such that $i < (s(c)-s(b))+1$ holds I is closed on $(\mathrm{StepForUp}(a,b,c,I,s))(i)$ and halting on $(\mathrm{StepForUp}(a,b,c,I,s))(i)$.

We now state several propositions:

(23) For every parahalting macro instruction I holds ProperForUpBody a_1, b_1, c_1, I, s.

(24) If $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(\mathrm{intloc}(0)) = 1$ and I_1 is closed on $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)$ and halting on $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)$, then $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k+1)(\mathrm{intloc}(0)) = 1$.

(25) Suppose $s(\mathrm{intloc}(0)) = 1$ and ProperForUpBody a, b_1, c_1, I_1, s. Let given k. Suppose $k \leq (s(c_1)-s(b_1))+1$. Then

(i) $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(\mathrm{intloc}(0)) = 1$,

(ii) if I_1 does not destroy a, then $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(a) = k+s(b_1)$ and $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(a) \leq s(c_1)+1$, and

(iii) $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(1^{\rm st}\text{-RWNotIn}(\{a,b_1,c_1\}\cup\mathrm{UsedIntLoc}(I_1)))+k = (s(c_1)-s(b_1))+1$.

(26) Suppose $s(\mathrm{intloc}(0)) = 1$ and ProperForUpBody a, b_1, c_1, I_1, s. Let given k. Then $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k)(1^{\rm st}\text{-RWNotIn}(\{a,b_1,c_1\}\cup\mathrm{UsedIntLoc}(I_1))) > 0$ if and only if $k < (s(c_1)-s(b_1))+1$.

(27) Suppose $s(\mathrm{intloc}(0)) = 1$ and ProperForUpBody a, b_1, c_1, I_1, s and $k < (s(c_1)-s(b_1))+1$. Then $(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k+1)\restriction(\{a,b_1,c_1\}\cup\mathrm{UsedIntLoc}(I_1)\cup F_2) = \mathrm{IExec}(I_1;\ \mathrm{AddTo}(a,\mathrm{intloc}(0)),(\mathrm{StepForUp}(a,b_1,c_1,I_1,s))(k))\restriction(\{a,b_1,c_1\}\cup\mathrm{UsedIntLoc}(I_1)\cup F_2)$, where $F_2 = \text{FinSeq-Locations}$.

Let a, b, c be integer locations and let I be a macro instruction. The functor $\text{for-up}(a,b,c,I)$ yields a macro instruction and is defined by:

(Def. 8) for-up(a,b,c,I) $=$ $(a_2{:=}c)$; SubFrom(a_2,b); AddTo$(a_2,\mathrm{intloc}(0))$; $(a{:=}b)$; (**while** $a_2 >$ 0 **do** (I; AddTo$(a,\mathrm{intloc}(0))$; SubFrom$(a_2,\mathrm{intloc}(0))$))), where $a_2 = 1^{\mathrm{st}}$-RWNotIn$(\{a,b,c\} \cup$ UsedIntLoc$(I))$.

The following proposition is true

(28) $\{a_1,b_1,c_1\} \cup$ UsedIntLoc$(I) \subseteq$ UsedIntLoc(for-up(a_1,b_1,c_1,I)).

Let a be a read-write integer location, let b, c be integer locations, and let I be a good macro instruction. One can check that for-up(a,b,c,I) is good.

We now state four propositions:

(29) If $a \neq a_1$ and $a_1 \neq 1^{\mathrm{st}}$-RWNotIn$(\{a,b_1,c_1\} \cup$ UsedIntLoc$(I))$ and I does not destroy a_1, then for-up(a,b_1,c_1,I) does not destroy a_1.

(30) Suppose $s(\mathrm{intloc}(0)) = 1$ and $s(b_1) > s(c_1)$. Then for every x such that $x \neq a$ and $x \in \{b_1,c_1\} \cup$ UsedIntLoc(I) holds (IExec(for-up$(a,b_1,c_1,I),s$))$(x) = s(x)$ and for every f holds (IExec(for-up$(a,b_1,c_1,I),s$))$(f) = s(f)$.

(31) Suppose $s(\mathrm{intloc}(0)) = 1$ but $k = (s(c_1) - s(b_1)) + 1$ but ProperForUpBody a, b_1, c_1, I_1, s or I_1 is parahalting. Then IExec(for-up$(a,b_1,c_1,I_1),s$)$\restriction D =$ (StepForUp(a,b_1,c_1,I_1,s))$(k)\restriction D$, where $D =$ Int-Locations $\cup$ FinSeq-Locations.

(32) Suppose $s(\mathrm{intloc}(0)) = 1$ but ProperForUpBody a, b_1, c_1, I_1, s or I_1 is parahalting. Then for-up(a,b_1,c_1,I_1) is closed on s and for-up(a,b_1,c_1,I_1) is halting on s.

4. Finding Minimum in a Section of an Array

Let s_1, f_1, m_2 be integer locations and let f be a finite sequence location. The functor FinSeqMin(f,s_1,f_1,m_2) yielding a macro instruction is defined by:

(Def. 9) FinSeqMin$(f,s_1,f_1,m_2) = (m_2{:=}s_1)$; for-up$(c_2,s_1,f_1,(a_3{:=}f_{c_2})$; $(a_4{:=}f_{m_2})$; (**if** $a_4 > a_3$ **then** Macro$(m_2{:=}c_2)$ **else** where $c_2 = 3^{\mathrm{rd}}$-RWNotIn$(\{s_1,f_1,m_2\})$, $a_3 = 1^{\mathrm{st}}$-RWNotIn$(\{s_1,f_1,m_2\})$, and $a_4 =$ 2^{nd}-RWNotIn$(\{s_1,f_1,m_2\})$.

Let s_1, f_1 be integer locations, let m_2 be a read-write integer location, and let f be a finite sequence location. Observe that FinSeqMin(f,s_1,f_1,m_2) is good.

We now state several propositions:

(33) If $c \neq a_1$, then FinSeqMin(f,a_1,b_1,c) does not destroy a_1.

(34) $\{a_1,b_1,c\} \subseteq$ UsedIntLoc(FinSeqMin(f,a_1,b_1,c)).

(35) If $s(\mathrm{intloc}(0)) = 1$, then FinSeqMin$(f,a_1,b_1,c)$ is closed on s and FinSeqMin(f,a_1,b_1,c) is halting on s.

(36) If $a_1 \neq c$ and $b_1 \neq c$ and $s(\mathrm{intloc}(0)) = 1$, then (IExec(FinSeqMin$(f,a_1,b_1,c),s$))$(f) =$ $s(f)$ and (IExec(FinSeqMin$(f,a_1,b_1,c),s$))$(a_1) = s(a_1)$ and (IExec(FinSeqMin$(f,a_1,b_1,c),s$))$(b_1) =$ $s(b_1)$.

(37) If $1 \leq s(a_1)$ and $s(a_1) \leq s(b_1)$ and $s(b_1) \leq \mathrm{len}\, s(f)$ and $a_1 \neq c$ and $b_1 \neq c$ and $s(\mathrm{intloc}(0)) =$ 1, then (IExec(FinSeqMin$(f,a_1,b_1,c),s$))$(c) = \min_{|s(a_1)|}^{|s(b_1)|} s(f)$.

5. A Swap Macro Instruction

Let f be a finite sequence location and let a, b be integer locations. The functor $\mathrm{swap}(f,a,b)$ yields a macro instruction and is defined by:

(Def. 10) $\mathrm{swap}(f,a,b) = (a_3{:=}f_a);\ (a_4{:=}f_b);\ (f_a{:=}a_4);\ (f_b{:=}a_3)$, where $a_3 = 1^{\mathrm{st}}\text{-RWNotIn}(\{a,b\})$ and $a_4 = 2^{\mathrm{nd}}\text{-RWNotIn}(\{a,b\})$.

Let f be a finite sequence location and let a, b be integer locations. Observe that $\mathrm{swap}(f,a,b)$ is good and parahalting.

One can prove the following propositions:

(38) If $c_1 \neq 1^{\mathrm{st}}\text{-RWNotIn}(\{a_1,b_1\})$ and $c_1 \neq 2^{\mathrm{nd}}\text{-RWNotIn}(\{a_1,b_1\})$, then $\mathrm{swap}(f,a_1,b_1)$ does not destroy c_1.

(39) If $1 \leq s(a_1)$ and $s(a_1) \leq \mathrm{len}\, s(f)$ and $1 \leq s(b_1)$ and $s(b_1) \leq \mathrm{len}\, s(f)$ and $s(\mathrm{intloc}(0)) = 1$, then $(\mathrm{IExec}(\mathrm{swap}(f,a_1,b_1),s))(f) = s(f) +\cdot (s(a_1), s(f)(s(b_1))) +\cdot (s(b_1), s(f)(s(a_1)))$.

(40) Suppose $1 \leq s(a_1)$ and $s(a_1) \leq \mathrm{len}\, s(f)$ and $1 \leq s(b_1)$ and $s(b_1) \leq \mathrm{len}\, s(f)$ and $s(\mathrm{intloc}(0)) = 1$. Then $(\mathrm{IExec}(\mathrm{swap}(f,a_1,b_1),s))(f)(s(a_1)) = s(f)(s(b_1))$ and $(\mathrm{IExec}(\mathrm{swap}(f,a_1,b_1),s))(f)(s(b_1)) = s(f)(s(a_1))$.

(41) $\{a_1,b_1\} \subseteq \mathrm{UsedIntLoc}(\mathrm{swap}(f,a_1,b_1))$.

(42) $\mathrm{UsedInt}^{*}\mathrm{Loc}(\mathrm{swap}(f,a_1,b_1)) = \{f\}$.

6. Selection Sort

Let f be a finite sequence location. The functor Selection-sort f yielding a macro instruction is defined as follows:

(Def. 11) $\text{Selection-sort}\, f = (f_1{:=}\mathrm{len}\, f);\ \text{for-up}(c_2, \mathrm{intloc}(0), f_1, \mathrm{FinSeqMin}(f,c_2,f_1,m_2);\ \mathrm{swap}(f,c_2,m_2))$, where $f_1 = 1^{\mathrm{st}}\text{-NotUsed}(\mathrm{swap}(f,c_2,m_2))$, $c_2 = 1^{\mathrm{st}}\text{-RWNotIn}(\emptyset_{\text{Int-Locations}})$, and $m_2 = 2^{\mathrm{nd}}\text{-RWNotIn}(\emptyset_{\text{Int-Locations}})$.

We now state the proposition

(43) Let S be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $S = \mathrm{IExec}(\text{Selection-sort}\, f, s)$. Then $S(f)$ is non decreasing on 1, $\mathrm{len}\, S(f)$ and there exists a permutation p of $\mathrm{dom}\, s(f)$ such that $S(f) = s(f) \cdot p$.

References

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

[2] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. 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. The ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal1.html.

[8] Grzegorz Bancerek. Sequences of ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal2.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 Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finseq_1.html.

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

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

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

[14] Czesław Byliński. Partial functions. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/partfun1.html.

[15] 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.

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

[17] Agata Darmochwał. Finite sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finset_1.html.

[18] Jarosław Kotowicz. Convergent real sequences. Upper and lower bound of sets of real numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/seq_4.html.

[19] Yatsuka Nakamura and Piotr Rudnicki. Vertex sequences induced by chains. *Journal of Formalized Mathematics*, 7, 1995. http://mizar.org/JFM/Vol7/graph_2.html.

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

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

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

[23] Andrzej Trybulec. Semilattice operations on finite subsets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/setwiseo.html.

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

[25] Andrzej Trybulec. On the sets inhabited by numbers. *Journal of Formalized Mathematics*, 15, 2003. http://mizar.org/JFM/Vol16/membered.html.

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

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

[28] 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.

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

[30] 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.

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

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

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

[34] Wojciech A. Trybulec. Pigeon hole principle. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/finseq_4.html.

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

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

[37] Edmund Woronowicz. Relations defined on sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/relset_1.html.

Received June 4, 1998

Published January 2, 2004