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

The `for` (going up) Macro Instruction

Piotr Rudnicki[1]
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 $\mathbf{SCM}_{\mathrm{FSA}}$. On the way, a macro for finding a minimum in a section of an array is defined.

MML Identifier: `SFMASTR3`.

The terminology and notation used in this paper have been introduced in the following articles: [16], [21], [28], [6], [7], [9], [26], [10], [11], [8], [25], [15], [5], [13], [29], [30], [23], [3], [4], [2], [1], [24], [22], [12], [19], [17], [18], [27], [20], and [14].

1. General Preliminaries

The following propositions are true:

(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 X be a finite non empty subset of $\mathbb{R}$. The functor $\min X$ yielding a real number is defined by:

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

ISSN 1426–2630

(Def. 1) $\min X \in X$ and for every real number k such that $k \in X$ holds $\min X \leqslant k$.

Let X be a finite non empty subset of $\mathbb{Z}$. The functor $\min X$ yielding an integer is defined by:

(Def. 2) There exists a finite non empty subset Y of $\mathbb{R}$ such that $Y = X$ and $\min X = \min Y$.

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

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

We use the following convention: F, F_1 denote finite sequences of elements of $\mathbb{Z}$ and k, m, n, m_1 denote natural numbers.

The following propositions are true:

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

(i) $m \leqslant m_1$,

(ii) $m_1 \leqslant n$,

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

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

(4) If $1 \leqslant m$ and $m \leqslant \operatorname{len} F$, 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 $m \leqslant i$ and $i \leqslant j$ and $j \leqslant n$ holds $F(i) \leqslant F(j)$.

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 $1 \leqslant i$ and $i \leqslant n$ and $n < j$ and $j \leqslant \operatorname{len} F$ holds $F(i) \leqslant F(j)$.

We now state two propositions:

(5) Suppose $k + 1 \leqslant \operatorname{len} F$ 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 $k + 1 \leqslant \operatorname{len} F$ 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 use the following convention: 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 $\mathrm{IExec}(\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, s){\restriction}D = s{\restriction}D$, where $D = \mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}$.

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

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

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

(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.

One can prove the following propositions:

(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 $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{StepForUp}(a, b, c, I, s)$ yields a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) and is defined by:

(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 $a_2 = 1^{\mathrm{st}}\text{-}\mathrm{RWNotIn}(\{a, b, c\} \cup \mathrm{UsedIntLoc}(I))$.

Next we 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 $(\text{StepForUp}(a, b_1, c_1, I, s))(0)(b_1) = s(b_1)$.

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

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

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

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

Let a, b, c be integer locations, let I be a macro instruction, and let s be a state of $\mathbf{SCM}_{\text{FSA}}$. 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 $(\text{StepForUp}(a, b, c, I, s))(i)$ and halting on $(\text{StepForUp}(a, b, c, I, s))(i)$.

Next we state several propositions:

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

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

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

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

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

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

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

(27) Suppose $s(\text{intloc}(0)) = 1$ and ProperForUpBody a, b_1, c_1, I_1, s and $k < (s(c_1)-s(b_1))+1$. Then $(\text{StepForUp}(a, b_1, c_1, I_1, s))(k+1)\restriction(\{a, b_1, c_1\} \cup \text{UsedIntLoc}(I_1) \cup F_2) = \text{IExec}(I_1;\text{AddTo}(a, \text{intloc}(0)), (\text{StepForUp}(a, b_1, c_1, I_1, s))(k))\restriction(\{a, b_1, c_1\} \cup \text{UsedIntLoc}(I_1) \cup F_2)$, 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) $\text{for-up}(a, b, c, I) =$
$(a_2{:=}c)$;
$\text{SubFrom}(a_2, b)$;
$\text{AddTo}(a_2, \text{intloc}(0))$;

$(a{:=}b)$;(**while** $a_2 > 0$ **do** (I;
AddTo(a, intloc(0));SubFrom(a_2, 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. Note that for-up(a, b, c, I) is good.

Next we 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(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(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(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** ($\mathrm{Stop}_{\mathrm{SCM_{FSA}}}$))),
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
$a_4 = 2^{\mathrm{nd}}$-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. Note that FinSeqMin(f, s_1, f_1, m_2) is good.

The following propositions are true:

(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 $1 \leqslant s(a_1)$ and $s(a_1) \leqslant s(b_1)$ and $s(b_1) \leqslant \mathrm{len}\, s(f)$ 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 swap(f, a, b) yields a macro instruction and is defined as follows:

(Def. 10) swap(f, a, b) = $(a_3{:=}f_a)$;$(a_4{:=}f_b)$;$(f_a{:=}a_4)$;$(f_b{:=}a_3)$, where $a_3 = 1^{\mathrm{st}}$-RWNotIn$(\{s_1, f_1, m_2\})$ and $a_4 = 2^{\mathrm{nd}}$-RWNotIn$(\{s_1, f_1, m_2\})$.

Let f be a finite sequence location and let a, b be integer locations. Note that swap(f, a, b) is good and parahalting.

The following propositions are true:

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

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

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

(42) UsedInt* Loc(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) Selection-sort f = $(f_1{:=}\mathrm{len} f)$; for-up$(c_2, \mathrm{intloc}(0), f'_1$, FinSeqMin$(f, c_2, f'_1, m'_1)$; swap$(f, c_2, m'_1))$, where $c_2 = 3^{\mathrm{rd}}$-RWNotIn$(\{s_1, f_1, m_2\})$, $f'_1 = 1^{\mathrm{st}}$-NotUsed(swap(f, c_2, m'_1)), and $m'_1 = 2^{\mathrm{nd}}$-RWNotIn$(\emptyset_{\text{Int-Locations}})$.

The following proposition is true

(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 $\mathrm{Seg}\, \mathrm{len}\, s(f)$ such that $S(f) = s(f) \cdot p$.

References

[1] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):73–80, 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, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Formalized Mathematics*, 6(**1**):41–47, 1997.

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

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

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

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

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

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

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

[12] Jing-Chao Chen. While macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**4**):553–561, 1997.

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

[14] Andrzej Kondracki. The chinese remainder theorem. *Formalized Mathematics*, 6(**4**):573–577, 1997.

[15] Rafał Kwiatek and Grzegorz Zwara. The divisibility of integers and integer relative primes. *Formalized Mathematics*, 1(**5**):829–832, 1990.

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

[17] Piotr Rudnicki. On the composition of non-parahalting macro instructions. *Formalized Mathematics*, 7(**1**):87–92, 1998.

[18] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.

[19] Andrzej Trybulec. Semilattice operations on finite subsets. *Formalized Mathematics*, 1(**2**):369–376, 1990.

[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 Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.

[24] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.

[25] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.

[26] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.

[27] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[28] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[29] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[30] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received June 4, 1998