FORMALIZED MATHEMATICS
Volume 8, Number 1, 1999
University of Białystok

The Construction and Shiftability of Program Blocks for SCMPDS[1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. In this article,a program block is defined as a finite sequence of instructions stored consecutively on initial positions. Based on this definition,any program block with more than two instructions can be viewed as the combination of two smaller program blocks. To describe the computation of a program block by the result of its two sub-blocks, we introduce the notions of paraclosed, parahalting, valid, and shiftable, the meaning of which may be stated as follows:

- a program is paraclosed if and only if any state containing it is closed,
- a program is parahalting if and only if any state containing it is halting,
- in a program block, a jumping instruction is valid if its jumping offset is valid,
- a program block is shiftable if it does not contain any return and saveIC instructions,and each instruction in it is valid.

When a program block is shiftable, its computing result does not depend on its storage position.

MML Identifier: `SCMPDS_4`.

The articles [17], [23], [12], [24], [5], [6], [20], [22], [2], [4], [11], [7], [13], [14], [18], [15], [3], [10], [9], [21], [19], [8], [1], and [16] provide the notation and terminology for this paper.

[1]This research is partially supported by the National Natural Science Foundation of China Grant No. 69873033.

ISSN 1426–2630

1. Definition of a Program Block and its Basic Properties

A Program-block is an initial programmed finite partial state of SCMPDS.

We adopt the following convention: m, n are natural numbers, i, j, k are instructions of SCMPDS, and I, J, K are Program-block.

Let us consider i. The functor Load(i) yielding a Program-block is defined as follows:

(Def. 1) $\mathrm{Load}(i) = \mathrm{inspos}\,0 \longmapsto i$.

Let us consider i. Note that Load(i) is non empty.

Next we state the proposition

(1) For every Program-block P and for every n holds $n < \mathrm{card}\,P$ iff $\mathrm{inspos}\,n \in \mathrm{dom}\,P$.

Let I be an initial finite partial state of SCMPDS. Note that ProgramPart(I) is initial.

Next we state four propositions:

(2) $\mathrm{dom}\,I$ misses $\mathrm{dom}\,\mathrm{Shift}(J, \mathrm{card}\,I)$.

(3) For every programmed finite partial state I of SCMPDS holds $\mathrm{card}\,\mathrm{Shift}(I, m) = \mathrm{card}\,I$.

(4) For all finite partial states I, J of SCMPDS holds $\mathrm{ProgramPart}(I{+\cdot}J) = \mathrm{ProgramPart}(I){+\cdot}\,\mathrm{ProgramPart}(J)$.

(5) For all finite partial states I, J of SCMPDS holds $\mathrm{Shift}(\mathrm{ProgramPart}(I{+\cdot}J), n) = \mathrm{Shift}(\mathrm{ProgramPart}(I), n){+\cdot}\,\mathrm{Shift}(\mathrm{ProgramPart}(J), n)$.

We use the following convention: a, b are Int position, s, s_1, s_2 are states of SCMPDS, and k_1, k_2 are integers.

Let us consider I. The functor Initialized(I) yields a finite partial state of SCMPDS and is defined as follows:

(Def. 2) $\mathrm{Initialized}(I) = I{+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{inspos}\,0)$.

We now state a number of propositions:

(6) $\mathrm{InsCode}(i) \in \{0, 1, 4, 5, 6\}$ or $(\mathrm{Exec}(i, s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$.

(7) $\mathbf{IC}_{\mathrm{SCMPDS}} \in \mathrm{dom}\,\mathrm{Initialized}(I)$.

(8) $\mathbf{IC}_{\mathrm{Initialized}(I)} = \mathrm{inspos}\,0$.

(9) $I \subseteq \mathrm{Initialized}(I)$.

(10) s and $s{+\cdot}I$ are equal outside the instruction locations of SCMPDS.

(11) Let s_1, s_2 be states of SCMPDS. Suppose $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and for every Int position a holds $s_1(a) = s_2(a)$. Then s_1 and s_2 are equal outside the instruction locations of SCMPDS.

(13)[2] Suppose s_1 and s_2 are equal outside the instruction locations of SCMPDS. Let a be a Int position. Then $s_1(a) = s_2(a)$.

(14) If s_1 and s_2 are equal outside the instruction locations of SCMPDS, then $s_1(\mathrm{DataLoc}(s_1(a), k_1)) = s_2(\mathrm{DataLoc}(s_2(a), k_1))$.

(15) Suppose s_1 and s_2 are equal outside the instruction locations of SCMPDS. Then $\mathrm{Exec}(i, s_1)$ and $\mathrm{Exec}(i, s_2)$ are equal outside the instruction locations of SCMPDS.

(16) Initialized$(I)\restriction$the instruction locations of SCMPDS $= I$.

(17) For all natural numbers k_1, k_2 such that $k_1 \neq k_2$ holds $\mathrm{DataLoc}(k_1, 0) \neq \mathrm{DataLoc}(k_2, 0)$.

(18) For every Int position d_1 there exists a natural number i such that $d_1 = \mathrm{DataLoc}(i, 0)$.

The scheme *SCMPDSEx* deals with a unary functor $\mathcal{F}$ yielding an instruction of SCMPDS, a unary functor $\mathcal{G}$ yielding an integer, and an instruction-location $\mathcal{A}$ of SCMPDS, and states that:

> There exists a state S of SCMPDS such that $\mathbf{IC}_S = \mathcal{A}$ and for every natural number i holds $S(\mathrm{inspos}\, i) = \mathcal{F}(i)$ and $S(\mathrm{DataLoc}(i, 0)) = \mathcal{G}(i)$

for all values of the parameters.

Next we state a number of propositions:

(19) For every state s of SCMPDS holds $\mathrm{dom}\, s = \{\mathbf{IC}_{\mathrm{SCMPDS}}\} \cup$ Data-$\mathrm{Loc}_{\mathrm{SCM}} \cup$ the instruction locations of SCMPDS.

(20) Let s be a state of SCMPDS and x be a set. Suppose $x \in \mathrm{dom}\, s$. Then x is a Int position or $x = \mathbf{IC}_{\mathrm{SCMPDS}}$ or x is an instruction-location of SCMPDS.

(21) Let s_1, s_2 be states of SCMPDS. Then for every instruction-location l of SCMPDS holds $s_1(l) = s_2(l)$ if and only if $s_1\restriction$the instruction locations of SCMPDS $= s_2\restriction$the instruction locations of SCMPDS.

(22) For every instruction-location i of SCMPDS holds $i \notin$ Data-$\mathrm{Loc}_{\mathrm{SCM}}$.

(23) For all states s_1, s_2 of SCMPDS holds for every Int position a holds $s_1(a) = s_2(a)$ iff $s_1\restriction$Data-$\mathrm{Loc}_{\mathrm{SCM}} = s_2\restriction$Data-$\mathrm{Loc}_{\mathrm{SCM}}$.

(24) Let s_1, s_2 be states of SCMPDS. Suppose s_1 and s_2 are equal outside the instruction locations of SCMPDS. Then $s_1\restriction$Data-$\mathrm{Loc}_{\mathrm{SCM}} = s_2\restriction$Data-$\mathrm{Loc}_{\mathrm{SCM}}$.

(25) For all states s, s_3 of SCMPDS and for every set A holds $(s_3{+\cdot}s\restriction A)\restriction A = s\restriction A$.

(26) For all Program-block I, J holds I and J are equal outside the instruction locations of SCMPDS.

[2]The proposition (12) has been removed.

(27) For every Program-block I holds $\operatorname{dom}\text{Initialized}(I) = \operatorname{dom} I \cup \{\mathbf{IC}_{\text{SCMPDS}}\}$.

(28) For every Program-block I and for every set x such that $x \in \operatorname{dom}\text{Initialized}(I)$ holds $x \in \operatorname{dom} I$ or $x = \mathbf{IC}_{\text{SCMPDS}}$.

(29) For every Program-block I holds $(\text{Initialized}(I))(\mathbf{IC}_{\text{SCMPDS}}) = \text{inspos}\,0$.

(30) For every Program-block I holds $\mathbf{IC}_{\text{SCMPDS}} \notin \operatorname{dom} I$.

(31) For every Program-block I and for every Int position a holds $a \notin \operatorname{dom}\text{Initialized}(I)$.

In the sequel x denotes a set.

The following propositions are true:

(32) If $x \in \operatorname{dom} I$, then $I(x) = (I{+\cdot}\,\text{Start-At}(\text{inspos}\,n))(x)$.

(33) For every Program-block I and for every set x such that $x \in \operatorname{dom} I$ holds $I(x) = (\text{Initialized}(I))(x)$.

(34) For all Program-block I, J and for every state s of SCMPDS such that $\text{Initialized}(J) \subseteq s$ holds $s{+\cdot}\,\text{Initialized}(I) = s{+\cdot}I$.

(35) For all Program-block I, J and for every state s of SCMPDS such that $\text{Initialized}(J) \subseteq s$ holds $\text{Initialized}(I) \subseteq s{+\cdot}I$.

(36) Let I, J be Program-block and s be a state of SCMPDS. Then $s{+\cdot}\,\text{Initialized}(I)$ and $s{+\cdot}\,\text{Initialized}(J)$ are equal outside the instruction locations of SCMPDS.

2. Combining two Consecutive Blocks into One Program Block

Let I, J be Program-block. The functor $I;J$ yields a Program-block and is defined by:

(Def. 3) $I;J = I{+\cdot}\,\text{Shift}(J, \operatorname{card} I)$.

One can prove the following propositions:

(37) For all Program-block I, J and for every instruction-location l of SCMPDS such that $l \in \operatorname{dom} I$ holds $(I;J)(l) = I(l)$.

(38) For all Program-block I, J and for every instruction-location l of SCMPDS such that $l \in \operatorname{dom} J$ holds $(I;J)(l + \operatorname{card} I) = J(l)$.

(39) For all Program-block I, J holds $\operatorname{dom} I \subseteq \operatorname{dom}(I;J)$.

(40) For all Program-block I, J holds $I \subseteq I;J$.

(41) For all Program-block I, J holds $I{+\cdot}(I;J) = I;J$.

(42) For all Program-block I, J holds $\text{Initialized}(I){+\cdot}(I;J) = \text{Initialized}(I;J)$.

3. Combining a Block and a Instruction into One Program Block

Let us consider i, J. The functor $i;J$ yielding a Program-block is defined by:

(Def. 4) $i;J = \mathrm{Load}(i);J$.

Let us consider I, j. The functor $I;j$ yields a Program-block and is defined by:

(Def. 5) $I;j = I;\mathrm{Load}(j)$.

Let us consider i, j. The functor $i;j$ yielding a Program-block is defined as follows:

(Def. 6) $i;j = \mathrm{Load}(i);\mathrm{Load}(j)$.

The following propositions are true:

(43) $i;j = \mathrm{Load}(i);j$.

(44) $i;j = i;\mathrm{Load}(j)$.

(45) $\mathrm{card}(I;J) = \mathrm{card}\, I + \mathrm{card}\, J$.

(46) $(I;J);K = I;(J;K)$.

(47) $(I;J);k = I;(J;k)$.

(48) $(I;j);K = I;(j;K)$.

(49) $(I;j);k = I;(j;k)$.

(50) $(i;J);K = i;(J;K)$.

(51) $(i;J);k = i;(J;k)$.

(52) $(i;j);K = i;(j;K)$.

(53) $(i;j);k = i;(j;k)$.

(54) $\mathrm{dom}\, I \cap \mathrm{dom}\, \text{Start-At}(\mathrm{inspos}\, n) = \emptyset$.

(55) $\text{Start-At}(\mathrm{inspos}\, 0) \subseteq \text{Initialized}(I)$.

(56) If $I+\cdot \text{Start-At}(\mathrm{inspos}\, n) \subseteq s$, then $I \subseteq s$.

(57) If $\text{Initialized}(I) \subseteq s$, then $I \subseteq s$.

(58) $(I+\cdot \text{Start-At}(\mathrm{inspos}\, n))\restriction$the instruction locations of SCMPDS $= I$.

In the sequel l, l_1 denote instructions-locations of SCMPDS.

Next we state four propositions:

(59) $a \notin \mathrm{dom}\, \text{Start-At}(l)$.

(60) $l_1 \notin \mathrm{dom}\, \text{Start-At}(l)$.

(61) $a \notin \mathrm{dom}(I+\cdot \text{Start-At}(l))$.

(62) $s+\cdot I+\cdot \text{Start-At}(\mathrm{inspos}\, 0) = s+\cdot \text{Start-At}(\mathrm{inspos}\, 0)+\cdot I$.

Let s be a state of SCMPDS, let l_2 be a Int position, and let k be an integer. Then $s +\cdot (l_2, k)$ is a state of SCMPDS.

4. The Notions of Paraclosed, Parahalting and their Basic Properties

Let I be a Program-block. The functor $\operatorname{stop} I$ yielding a Program-block is defined as follows:

(Def. 7) $\operatorname{stop} I = I;\mathrm{SCMPDS} - \mathrm{Stop}$.

Let I be a Program-block and let s be a state of SCMPDS. The functor $\mathrm{IExec}(I, s)$ yielding a state of SCMPDS is defined as follows:

(Def. 8) $\mathrm{IExec}(I, s) = \mathrm{Result}(s{+\cdot}\operatorname{Initialized}(\operatorname{stop} I)){+\cdot}s{\restriction}$the instruction locations of SCMPDS.

Let I be a Program-block. We say that I is paraclosed if and only if:

(Def. 9) For every state s of SCMPDS and for every natural number n such that $\operatorname{Initialized}(\operatorname{stop} I) \subseteq s$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(n)} \in \operatorname{dom}\operatorname{stop} I$.

We say that I is parahalting if and only if:

(Def. 10) $\operatorname{Initialized}(\operatorname{stop} I)$ is halting.

Let us note that there exists a Program-block which is parahalting.

One can prove the following proposition

(63) For every parahalting Program-block I such that $\operatorname{Initialized}(\operatorname{stop} I) \subseteq s$ holds s is halting.

Let I be a parahalting Program-block. Note that $\operatorname{Initialized}(\operatorname{stop} I)$ is halting.

Let l_3, l_4 be instructions-locations of SCMPDS and let a, b be instructions of SCMPDS. Then $[l_3 \longmapsto a, l_4 \longmapsto b]$ is a finite partial state of SCMPDS.

One can prove the following propositions:

(64) For every integer k such that $k \neq 0$ holds goto $k \neq \mathbf{halt}_{\mathrm{SCMPDS}}$.

(65) $\mathbf{IC}_s \neq \mathrm{Next}(\mathbf{IC}_s)$.

(66) $s_2{+\cdot}[\mathbf{IC}_{(s_2)} \longmapsto \text{goto } 1, \mathrm{Next}(\mathbf{IC}_{(s_2)}) \longmapsto \text{goto } (-1)]$ is not halting.

(67) Suppose that

(i) s_1 and s_2 are equal outside the instruction locations of SCMPDS,

(ii) $I \subseteq s_1$,

(iii) $I \subseteq s_2$, and

(iv) for every m such that $m < n$ holds $\mathbf{IC}_{(\mathrm{Computation}(s_2))(m)} \in \operatorname{dom} I$.

Let given m. Suppose $m \leqslant n$. Then $(\mathrm{Computation}(s_1))(m)$ and $(\mathrm{Computation}(s_2))(m)$ are equal outside the instruction locations of SCMPDS.

(68) For every state s of SCMPDS and for every instruction-location l of SCMPDS holds $l \in \operatorname{dom} s$.

In the sequel l_1, l_5 are instructions-locations of SCMPDS and i_1, i_2 are instructions of SCMPDS.

The following propositions are true:

(69) $s+\cdot[l_1 \longmapsto i_1, l_5 \longmapsto i_2] = s +\cdot (l_1, i_1) +\cdot (l_5, i_2)$.

(70) $\mathrm{Next}(\mathrm{inspos}\, n) = \mathrm{inspos}\, n + 1$.

(71) If $\mathbf{IC}_s \notin \mathrm{dom}\, I$, then $\mathrm{Next}(\mathbf{IC}_s) \notin \mathrm{dom}\, I$.

Let us mention that every Program-block which is parahalting is also paraclosed.

We now state several propositions:

(72) $\mathrm{dom}\, \mathrm{SCMPDS} - \mathrm{Stop} = \{\mathrm{inspos}\, 0\}$.

(73) $\mathrm{inspos}\, 0 \in \mathrm{dom}\, \mathrm{SCMPDS} - \mathrm{Stop}$ and $(\mathrm{SCMPDS} - \mathrm{Stop})(\mathrm{inspos}\, 0) = \mathbf{halt}_{\mathrm{SCMPDS}}$.

(74) $\mathrm{card}\, \mathrm{SCMPDS} - \mathrm{Stop} = 1$.

(75) $\mathrm{inspos}\, 0 \in \mathrm{dom}\, \mathrm{stop}\, I$.

(76) Let p be a programmed finite partial state of SCMPDS, k be a natural number, and i_3 be an instruction-location of SCMPDS. If $i_3 \in \mathrm{dom}\, p$, then $i_3 + k \in \mathrm{dom}\, \mathrm{Shift}(p, k)$.

5. Shiftability of Program Blocks and Instructions

Let i be an instruction of SCMPDS and let n be a natural number. We say that i valid at n if and only if the conditions (Def. 11) are satisfied.

(Def. 11)(i) If $\mathrm{InsCode}(i) = 0$, then there exists k_1 such that $i = \mathrm{goto}\, k_1$ and $n + k_1 \geqslant 0$,

(ii) if $\mathrm{InsCode}(i) = 4$, then there exist a, k_1, k_2 such that $i = (a, k_1) <> 0\_gotok_2$ and $n + k_2 \geqslant 0$,

(iii) if $\mathrm{InsCode}(i) = 5$, then there exist a, k_1, k_2 such that $i = (a, k_1) <= 0\_gotok_2$ and $n + k_2 \geqslant 0$, and

(iv) if $\mathrm{InsCode}(i) = 6$, then there exist a, k_1, k_2 such that $i = (a, k_1) >= 0\_gotok_2$ and $n + k_2 \geqslant 0$.

One can prove the following proposition

(77) Let i be an instruction of SCMPDS and m, n be natural numbers. If i valid at m and $m \leqslant n$, then i valid at n.

Let I_1 be a finite partial state of SCMPDS. We say that I_1 is shiftable if and only if:

(Def. 12) For all n, i such that $\mathrm{inspos}\, n \in \mathrm{dom}\, I_1$ and $i = I_1(\mathrm{inspos}\, n)$ holds $\mathrm{InsCode}(i) \neq 1$ and $\mathrm{InsCode}(i) \neq 3$ and i valid at n.

Let us mention that there exists a Program-block which is parahalting and shiftable.

Let i be an instruction of SCMPDS. We say that i is shiftable if and only if:

(Def. 13) $\mathrm{InsCode}(i) = 2$ or $\mathrm{InsCode}(i) > 6$.

One can check that there exists an instruction of SCMPDS which is shiftable.

Let us consider a, k_1. Observe that $a{:=}k_1$ is shiftable.

Let us consider a, k_1, k_2. One can check that $a_{k_1}{:=}k_2$ is shiftable.

Let us consider a, k_1, k_2. Observe that $\mathrm{AddTo}(a, k_1, k_2)$ is shiftable.

Let us consider a, b, k_1, k_2. One can check the following observations:

* $\mathrm{AddTo}(a, k_1, b, k_2)$ is shiftable,
* $\mathrm{SubFrom}(a, k_1, b, k_2)$ is shiftable,
* $\mathrm{MultBy}(a, k_1, b, k_2)$ is shiftable,
* $\mathrm{Divide}(a, k_1, b, k_2)$ is shiftable, and
* $(a, k_1) := (b, k_2)$ is shiftable.

Let I, J be shiftable Program-block. Observe that $I;J$ is shiftable.

Let i be a shiftable instruction of SCMPDS. Observe that $\mathrm{Load}(i)$ is shiftable.

Let i be a shiftable instruction of SCMPDS and let J be a shiftable Program-block. Observe that $i;J$ is shiftable.

Let I be a shiftable Program-block and let j be a shiftable instruction of SCMPDS. Observe that $I;j$ is shiftable.

Let i, j be shiftable instructions of SCMPDS. Note that $i;j$ is shiftable.

Let us note that SCMPDS − Stop is parahalting and shiftable.

Let I be a shiftable Program-block. One can verify that $\mathrm{stop}\, I$ is shiftable.

Next we state the proposition

(78) For every shiftable Program-block I and for every integer k_1 such that $\mathrm{card}\, I + k_1 \geqslant 0$ holds $I;\mathrm{goto}\ k_1$ is shiftable.

Let n be a natural number. Note that $\mathrm{Load}(\mathrm{goto}\ n)$ is shiftable.

One can prove the following proposition

(79) Let I be a shiftable Program-block, k_1, k_2 be integers, and a be a Int position. If $\mathrm{card}\, I + k_2 \geqslant 0$, then $I;((a, k_1) <> 0\_goto k_2)$ is shiftable.

Let k_1 be an integer, let a be a Int position, and let n be a natural number. Note that $\mathrm{Load}((a, k_1) <> 0\_goto n)$ is shiftable.

Next we state the proposition

(80) Let I be a shiftable Program-block, k_1, k_2 be integers, and a be a Int position. If $\mathrm{card}\, I + k_2 \geqslant 0$, then $I;((a, k_1) <= 0\_goto k_2)$ is shiftable.

Let k_1 be an integer, let a be a Int position, and let n be a natural number. Observe that $\mathrm{Load}((a, k_1) <= 0\_goto n)$ is shiftable.

One can prove the following proposition

(81) Let I be a shiftable Program-block, k_1, k_2 be integers, and a be a Int position. If $\mathrm{card}\, I + k_2 \geqslant 0$, then $I;((a, k_1) >= 0\_goto k_2)$ is shiftable.

Let k_1 be an integer, let a be a Int position, and let n be a natural number. Observe that $\mathrm{Load}((a, k_1) >= 0\_goto n)$ is shiftable.

We now state three propositions:

(82) Let s_1, s_2 be states of SCMPDS, n, m be natural numbers, and k_1 be an integer. If $\mathbf{IC}_{(s_1)} = \operatorname{inspos} m$ and $m + k_1 \geqslant 0$ and $\mathbf{IC}_{(s_1)} + n = \mathbf{IC}_{(s_2)}$, then $\operatorname{ICplusConst}(s_1, k_1) + n = \operatorname{ICplusConst}(s_2, k_1)$.

(83) Let s_1, s_2 be states of SCMPDS, n, m be natural numbers, and i be an instruction of SCMPDS. Suppose $\mathbf{IC}_{(s_1)} = \operatorname{inspos} m$ and i valid at m and $\operatorname{InsCode}(i) \neq 1$ and $\operatorname{InsCode}(i) \neq 3$ and $\mathbf{IC}_{(s_1)} + n = \mathbf{IC}_{(s_2)}$ and $s_1 \restriction \text{Data-Loc}_{\mathrm{SCM}} = s_2 \restriction \text{Data-Loc}_{\mathrm{SCM}}$. Then $\mathbf{IC}_{\operatorname{Exec}(i, s_1)} + n = \mathbf{IC}_{\operatorname{Exec}(i, s_2)}$ and $\operatorname{Exec}(i, s_1) \restriction \text{Data-Loc}_{\mathrm{SCM}} = \operatorname{Exec}(i, s_2) \restriction \text{Data-Loc}_{\mathrm{SCM}}$.

(84) Let J be a parahalting shiftable Program-block. Suppose $\operatorname{Initialized}(\operatorname{stop} J) \subseteq s_1$. Let n be a natural number. Suppose $\operatorname{Shift}(\operatorname{stop} J, n) \subseteq s_2$ and $\mathbf{IC}_{(s_2)} = \operatorname{inspos} n$ and $s_1 \restriction \text{Data-Loc}_{\mathrm{SCM}} = s_2 \restriction \text{Data-Loc}_{\mathrm{SCM}}$. Let i be a natural number. Then $\mathbf{IC}_{(\operatorname{Computation}(s_1))(i)} + n = \mathbf{IC}_{(\operatorname{Computation}(s_2))(i)}$ and $\operatorname{CurInstr}((\operatorname{Computation}(s_1))(i)) = \operatorname{CurInstr}((\operatorname{Computation}(s_2))(i))$ and $(\operatorname{Computation}(s_1))(i) \restriction \text{Data-Loc}_{\mathrm{SCM}} = (\operatorname{Computation}(s_2))(i) \restriction \text{Data-Loc}_{\mathrm{SCM}}$.

References

[1] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.

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

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

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

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

[6] 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.

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

[8] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Formalized Mathematics*, 8(**1**):193–199, 1999.

[9] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Formalized Mathematics*, 8(**1**):183–191, 1999.

[10] Jing-Chao Chen. A small computer model with push-down stack. *Formalized Mathematics*, 8(**1**):175–182, 1999.

[11] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.

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

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

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

[15] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.

[16] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.

[17] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.

[18] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.

[19] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[20] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[21] Wojciech A. Trybulec. Groups. *Formalized Mathematics*, 1(**5**):821–827, 1990.
[22] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[23] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[24] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received June 15, 1999