JOURNAL OF FORMALIZED MATHEMATICS
Volume 11, Released 1999, Published 2003
Inst. of Computer Science, Univ. 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`.

WWW: `http://mizar.org/JFM/Vol11/scmpds_4.html`

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

1. Definition of a Program Block and its Basic Properties

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

We follow the rules: m, n are natural numbers, i, j, k are instructions of SCMPDS, and I, J, K are Program-blocks.

Let us consider i. The functor $\mathrm{Load}(i)$ yielding a Program-block is defined by:

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

Let us consider i. One can check that $\mathrm{Load}(i)$ is non empty.

We now 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. Observe that $\mathrm{ProgramPart}(I)$ is initial.

The following propositions are true:

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

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

The following propositions are true:

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

(11)[1] 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 an 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) $\mathrm{Initialized}(I) \upharpoonright$ 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 $\mathrm{dom}\, s = \{\mathbf{IC}_{\mathrm{SCMPDS}}\} \cup \text{Data-Loc}_{\mathrm{SCM}} \cup$ 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 an 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 $s_1 \upharpoonright$ the instruction locations of SCMPDS $= s_2 \upharpoonright$ the instruction locations of SCMPDS.

[1] The proposition (10) has been removed.
[2] The proposition (12) has been removed.

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

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

(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 $s_1{\restriction}\text{Data-Loc}_{\text{SCM}} = s_2{\restriction}\text{Data-Loc}_{\text{SCM}}$.

(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-blocks I, J holds I and J are equal outside the instruction locations of SCMPDS.

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

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

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

In the sequel x denotes a set.
One can prove the following propositions:

(32) If $x \in \text{dom}\,I$, 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 $x \in \text{dom}\,I$ holds $I(x) = (\text{Initialized}(I))(x)$.

(34) For all Program-blocks 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-blocks 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-blocks 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-blocks. The functor $I; J$ yields a Program-block and is defined as follows:

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

We now state several propositions:

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

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

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

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

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

(42) For all Program-blocks 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 by:

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

One can prove the following propositions:

(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) $\mathrm{dom}\,I$ misses $\mathrm{dom}\,\mathrm{Start\text{-}At}(\mathrm{inspos}\,n)$.

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

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

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

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

In the sequel l, l_1 are instruction-locations of SCMPDS.
Next we state four propositions:

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

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

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

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

Let s be a state of SCMPDS, let l_2 be an 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 $\mathrm{stop}\, I$ yielding a Program-block is defined as follows:

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

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 by:

(Def. 8) $\mathrm{IExec}(I,s) = \mathrm{Result}(s + \cdot\, \mathrm{Initialized}(\mathrm{stop}\, I)) + \cdot\, s \restriction$ 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 $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(n)} \in \mathrm{dom}\, \mathrm{stop}\, I$.

We say that I is parahalting if and only if:

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

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

We now state the proposition

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

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

Let l_3, l_4 be instruction-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 four propositions:

(65)[3] $\mathbf{IC}_s \neq \mathrm{Next}(\mathbf{IC}_s)$.

(66) $s_2 + \cdot\, [\mathbf{IC}_{(s_2)} \longmapsto \mathrm{goto}\ 1, \mathrm{Next}(\mathbf{IC}_{(s_2)}) \longmapsto \mathrm{goto}\ (-1)]$ 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 $\mathbf{IC}_{(\mathrm{Computation}(s_2))(m)} \in \mathrm{dom}\, I$.

Let given m. Suppose $m \leq n$. 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 $l \in \mathrm{dom}\, s$.

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

Next we state three propositions:

(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 observe that every Program-block which is parahalting is also paraclosed.

The following propositions are true:

(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 $i_3 + k \in \mathrm{dom}\, \mathrm{Shift}(p,k)$.

[3] The proposition (64) has been removed.

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 $n + k_1 \geq 0$,

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

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

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

We now state the proposition

(77) Let i be an instruction of SCMPDS and m, n be natural numbers. If i valid at m and $m \leq n$, 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 $\mathrm{InsCode}(i) > 6$.

Let us note that there exists an instruction of SCMPDS which is shiftable.
Let us consider a, k_1. Note that $a{:=}k_1$ is shiftable.
Let us consider a, k_1, k_2. Note 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-blocks. Note that I; J is shiftable.
Let i be a shiftable instruction of SCMPDS. Note 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. One can verify that I; j is shiftable.
Let i, j be shiftable instructions of SCMPDS. Observe that i; j is shiftable.
Let us note that SCMPDS $-$ Stop is parahalting and shiftable.
Let I be a shiftable Program-block. Note that $\mathrm{stop}\, I$ is shiftable.
We now state the proposition

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

Let n be a natural number. One can verify that $\mathrm{Load}(\mathrm{goto}\, n)$ is shiftable.
Next we state the proposition

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

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

One can prove the following proposition

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

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

The following proposition is true

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

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

One can prove the following three propositions:

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

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

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

REFERENCES

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

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

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

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

[5] Czesław Byliński. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/cqc_lang.html.

[6] 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.

[7] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Journal of Formalized Mathematics*, 11, 1999. http://mizar.org/JFM/Vol11/scmpds_3.html.

[8] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Journal of Formalized Mathematics*, 11, 1999. http://mizar.org/JFM/Vol11/scmpds_2.html.

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

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

[11] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Journal of Formalized Mathematics*, 4, 1992. http://mizar.org/JFM/Vol4/ami_2.html.

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

[13] Andrzej Trybulec. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/enumset1.html.

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

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

[16] 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.

[17] 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.

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

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

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

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

Received June 15, 1999

Published January 2, 2004