JOURNAL OF FORMALIZED MATHEMATICS
Volume 5, Released 1993, Published 2003
Inst. of Computer Science, Univ. of Białystok

On the Decomposition of the States of SCM

Yasushi Tanaka
Shinshu University
Information Engineering Dept.
Nagano

Summary. This article continues the development of the basic terminology for the **SCM** as defined in [11],[12], [19]. There is developed of the terminology for discussing static properties of instructions (i.e. not related to execution), for data locations, instruction locations, as well as for states and partial states of **SCM**. The main contribution of the article consists in characterizing **SCM** computations starting in states containing autonomic finite partial states.

MML Identifier: `AMI_5`.

WWW: `http://mizar.org/JFM/Vol5/ami_5.html`

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

1. Preliminaries

One can prove the following propositions:

(3)[1] For all natural numbers m, k such that $k \neq 0$ holds $m \cdot k \div k = m$.

(4) For all natural numbers i, j such that $i \geq j$ holds $(i -' j) + j = i$.

(5) For all functions f, g and for all sets A, B such that $A \subseteq B$ and $f{\restriction}B = g{\restriction}B$ holds $f{\restriction}A = g{\restriction}A$.

(6) For all functions p, q and for every set A holds $(p{+\cdot}q){\restriction}A = p{\restriction}A{+\cdot}q{\restriction}A$.

(7) For all functions f, g and for every set A such that A misses $\operatorname{dom} g$ holds $(f{+\cdot}g){\restriction}A = f{\restriction}A$.

(8) For all functions f, g and for every set A such that $\operatorname{dom} f$ misses A holds $(f{+\cdot}g){\restriction}A = g{\restriction}A$.

(9) For all functions f, g, h such that $\operatorname{dom} g = \operatorname{dom} h$ holds $f{+\cdot}g{+\cdot}h = f{+\cdot}h$.

(10) For all functions f, g such that $f \subseteq g$ holds $f{+\cdot}g = g$ and $g{+\cdot}f = g$.

(11) For every function f and for every set A holds $f{+\cdot}f{\restriction}A = f$.

(12) For all functions f, g and for all sets B, C such that $\operatorname{dom} f \subseteq B$ and $\operatorname{dom} g \subseteq C$ and B misses C holds $(f{+\cdot}g){\restriction}B = f$ and $(f{+\cdot}g){\restriction}C = g$.

(13) For all functions p, q and for every set A such that $\operatorname{dom} p \subseteq A$ and $\operatorname{dom} q$ misses A holds $(p{+\cdot}q){\restriction}A = p$.

(14) For every function f and for all sets A, B holds $f{\restriction}(A \cup B) = f{\restriction}A{+\cdot}f{\restriction}B$.

[1] The propositions (1) and (2) have been removed.

2. Total states of SCM

The following propositions are true:

(15) Let a be a data-location and s be a state of **SCM**. Then $(\mathrm{Exec}(\mathrm{Divide}(a,a),s))(\mathbf{IC_{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{Divide}(a,a),s))(a) = s(a) \bmod s(a)$ and for every data-location c such that $c \neq a$ holds $(\mathrm{Exec}(\mathrm{Divide}(a,a),s))(c) = s(c)$.

(16) For every set x such that $x \in \mathrm{Data\text{-}Loc_{SCM}}$ holds x is a data-location.

(18)[2] For every data-location d_1 there exists a natural number i such that $d_1 = \mathbf{d}_i$.

(19) For every instruction-location i_1 of **SCM** there exists a natural number i such that $i_1 = \mathbf{i}_i$.

(20) For every data-location d_1 holds $d_1 \neq \mathbf{IC_{SCM}}$.

In the sequel N denotes a set with non empty elements and S denotes an IC-Ins-separated definite non empty non void AMI over N.

Next we state a number of propositions:

(22)[3] For every instruction-location i_1 of **SCM** and for every data-location d_1 holds $i_1 \neq d_1$.

(23) The carrier of $\mathbf{SCM} = \{\mathbf{IC_{SCM}}\} \cup \mathrm{Data\text{-}Loc_{SCM}} \cup \mathrm{Instr\text{-}Loc_{SCM}}$.

(24) Let s be a state of **SCM**, d be a data-location, and l be an instruction-location of **SCM**. Then $d \in \mathrm{dom}\, s$ and $l \in \mathrm{dom}\, s$.

(25) For every state s of S holds $\mathbf{IC}_S \in \mathrm{dom}\, s$.

(26) Let s_1, s_2 be states of **SCM**. Suppose $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and for every data-location a holds $s_1(a) = s_2(a)$ and for every instruction-location i of **SCM** holds $s_1(i) = s_2(i)$. Then $s_1 = s_2$.

(27) For every state s of **SCM** holds $\mathrm{Data\text{-}Loc_{SCM}} \subseteq \mathrm{dom}\, s$.

(28) For every state s of **SCM** holds $\mathrm{Instr\text{-}Loc_{SCM}} \subseteq \mathrm{dom}\, s$.

(29) For every state s of **SCM** holds $\mathrm{dom}(s{\restriction}\mathrm{Data\text{-}Loc_{SCM}}) = \mathrm{Data\text{-}Loc_{SCM}}$.

(30) For every state s of **SCM** holds $\mathrm{dom}(s{\restriction}\mathrm{Instr\text{-}Loc_{SCM}}) = \mathrm{Instr\text{-}Loc_{SCM}}$.

(31) $\mathrm{Data\text{-}Loc_{SCM}}$ is not finite.

(32) The instruction locations of **SCM** are not finite.

One can verify that $\mathrm{Data\text{-}Loc_{SCM}}$ is infinite and the instruction locations of **SCM** is infinite.

We now state three propositions:

(33) $\mathrm{Data\text{-}Loc_{SCM}}$ misses $\mathrm{Instr\text{-}Loc_{SCM}}$.

(34) For every state s of S holds $\mathrm{Start\text{-}At}(\mathbf{IC}_s) = s{\restriction}\{\mathbf{IC}_S\}$.

(35) For every instruction-location l of S holds $\mathrm{Start\text{-}At}(l) = \{\langle \mathbf{IC}_S, l\rangle\}$.

Let N be a set, let A be an AMI over N, and let I be an element of the instructions of A. The functor $\mathrm{InsCode}(I)$ yielding an instruction type of A is defined by:

(Def. 1) $\mathrm{InsCode}(I) = I_{\mathbf{1}}$.

Let I be an instruction of **SCM**. Observe that $\mathrm{InsCode}(I)$ is natural.

Let I be an instruction of **SCM**. The functor ${}^{@}I$ yielding an element of $\mathrm{Instr_{SCM}}$ is defined as follows:

[2] The proposition (17) has been removed.

[3] The proposition (21) has been removed.

(Def. 2) $^{@}I = I$.

Let l_1 be an element of Instr-Loc$_{\rm SCM}$. The functor $l_1{}^{\rm T}$ yields an instruction-location of **SCM** and is defined by:

(Def. 3) $l_1{}^{\rm T} = l_1$.

Let l_1 be an element of Data-Loc$_{\rm SCM}$. The functor $l_1{}^{\rm T}$ yielding a data-location is defined as follows:

(Def. 4) $l_1{}^{\rm T} = l_1$.

One can prove the following proposition

(36) For every instruction l of **SCM** holds $\mathrm{InsCode}(l) \leq 8$.

In the sequel a, b are data-locations and l_1 is an instruction-location of **SCM**.
The following propositions are true:

(37) $\mathrm{InsCode}(\mathbf{halt}_{\mathbf{SCM}}) = 0$.

(38) $\mathrm{InsCode}(a{:=}b) = 1$.

(39) $\mathrm{InsCode}(\mathrm{AddTo}(a,b)) = 2$.

(40) $\mathrm{InsCode}(\mathrm{SubFrom}(a,b)) = 3$.

(41) $\mathrm{InsCode}(\mathrm{MultBy}(a,b)) = 4$.

(42) $\mathrm{InsCode}(\mathrm{Divide}(a,b)) = 5$.

(43) $\mathrm{InsCode}(\mathrm{goto}\ l_1) = 6$.

(44) $\mathrm{InsCode}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1) = 7$.

(45) $\mathrm{InsCode}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1) = 8$.

In the sequel d_2, d_3 denote data-locations and l_1 denotes an instruction-location of **SCM**.
The following propositions are true:

(46) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 0$ holds $i_2 = \mathbf{halt}_{\mathbf{SCM}}$.

(47) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 1$ there exist d_2, d_3 such that $i_2 = d_2{:=}d_3$.

(48) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 2$ there exist d_2, d_3 such that $i_2 = \mathrm{AddTo}(d_2,d_3)$.

(49) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 3$ there exist d_2, d_3 such that $i_2 = \mathrm{SubFrom}(d_2,d_3)$.

(50) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 4$ there exist d_2, d_3 such that $i_2 = \mathrm{MultBy}(d_2,d_3)$.

(51) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 5$ there exist d_2, d_3 such that $i_2 = \mathrm{Divide}(d_2,d_3)$.

(52) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 6$ there exists l_1 such that $i_2 = \mathrm{goto}\ l_1$.

(53) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 7$ there exist l_1, d_2 such that $i_2 = \mathbf{if}\ d_2 = 0\ \mathbf{goto}\ l_1$.

(54) For every instruction i_2 of **SCM** such that $\mathrm{InsCode}(i_2) = 8$ there exist l_1, d_2 such that $i_2 = \mathbf{if}\ d_2 > 0\ \mathbf{goto}\ l_1$.

(55) For every instruction-location l_1 of **SCM** holds $(^{@}\text{goto } l_1)\,\text{address}_{\text{j}} = l_1$.

(56) For every instruction-location l_1 of **SCM** and for every data-location a holds $(^{@}(\textbf{if } a = 0 \textbf{ goto } l_1))\,\text{address}_{\text{j}} = l_1$ and $(^{@}(\textbf{if } a = 0 \textbf{ goto } l_1))\,\text{address}_{\text{c}} = a$.

(57) For every instruction-location l_1 of **SCM** and for every data-location a holds $(^{@}(\textbf{if } a > 0 \textbf{ goto } l_1))\,\text{address}_{\text{j}} = l_1$ and $(^{@}(\textbf{if } a > 0 \textbf{ goto } l_1))\,\text{address}_{\text{c}} = a$.

(58) For all states s_1, s_2 of **SCM** such that $s_1 \restriction (\text{Data-Loc}_{\text{SCM}} \cup \{\mathbf{IC}_{\mathbf{SCM}}\}) = s_2 \restriction (\text{Data-Loc}_{\text{SCM}} \cup \{\mathbf{IC}_{\mathbf{SCM}}\})$ and for every instruction l of **SCM** holds $\text{Exec}(l, s_1) \restriction (\text{Data-Loc}_{\text{SCM}} \cup \{\mathbf{IC}_{\mathbf{SCM}}\}) = \text{Exec}(l, s_2) \restriction (\text{Data-Loc}_{\text{SCM}} \cup \{\mathbf{IC}_{\mathbf{SCM}}\})$.

(59) For every instruction i of **SCM** and for every state s of **SCM** holds $\text{Exec}(i, s) \restriction \text{Instr-Loc}_{\text{SCM}} = s \restriction \text{Instr-Loc}_{\text{SCM}}$.

3. Finite partial states of **SCM**

Next we state the proposition

(60) For every finite partial state p of S and for every state s of S such that $\mathbf{IC}_S \in \operatorname{dom} p$ and $p \subseteq s$ holds $\mathbf{IC}_p = \mathbf{IC}_s$.

Let us consider N, S, let p be a finite partial state of S, and let l_1 be an instruction-location of S. Let us assume that $l_1 \in \operatorname{dom} p$. The functor $\pi_{l_1} p$ yields an instruction of S and is defined as follows:

(Def. 5) $\pi_{l_1} p = p(l_1)$.

Next we state the proposition

(61) Let N be a set, S be an AMI over N, x be a set, and p be a finite partial state of S. If $x \subseteq p$, then x is a finite partial state of S.

Let N be a set, let S be an AMI over N, and let p be a finite partial state of S. The functor $\text{ProgramPart}(p)$ yielding a programmed finite partial state of S is defined as follows:

(Def. 6) $\text{ProgramPart}(p) = p \restriction$the instruction locations of S.

Let N be a set, let S be a non empty AMI over N, and let p be a finite partial state of S. The functor $\text{DataPart}(p)$ yielding a finite partial state of S is defined by:

(Def. 7) $\text{DataPart}(p) = p \restriction ((\text{the carrier of } S) \setminus (\{\mathbf{IC}_S\} \cup \text{the instruction locations of } S))$.

Let N be a set, let S be a non empty AMI over N, and let I_1 be a finite partial state of S. We say that I_1 is data-only if and only if:

(Def. 8) $\operatorname{dom} I_1$ misses $\{\mathbf{IC}_S\} \cup$ the instruction locations of S.

Let N be a set and let S be a non empty AMI over N. Note that there exists a finite partial state of S which is data-only.

We now state a number of propositions:

(62) For every set N and for every non empty AMI S over N and for every finite partial state p of S holds $\text{DataPart}(p) \subseteq p$.

(63) For every set N and for every AMI S over N and for every finite partial state p of S holds $\text{ProgramPart}(p) \subseteq p$.

(64) Let S be a steady-programmed IC-Ins-separated definite non empty non void AMI over N, p be a finite partial state of S, and s be a state of S. If $p \subseteq s$, then for every natural number i holds $\text{ProgramPart}(p) \subseteq (\text{Computation}(s))(i)$.

(65) For every set N and for every non empty AMI S over N and for every finite partial state p of S holds $\mathbf{IC}_S \notin \operatorname{dom}\text{DataPart}(p)$.

(66) Let S be an IC-Ins-separated definite realistic non empty non void AMI over N and p be a finite partial state of S. Then $\mathbf{IC}_S \notin \mathrm{dom}\,\mathrm{ProgramPart}(p)$.

(67) For every set N and for every non empty AMI S over N and for every finite partial state p of S holds $\{\mathbf{IC}_S\}$ misses $\mathrm{dom}\,\mathrm{DataPart}(p)$.

(68) Let S be an IC-Ins-separated definite realistic non empty non void AMI over N and p be a finite partial state of S. Then $\{\mathbf{IC}_S\}$ misses $\mathrm{dom}\,\mathrm{ProgramPart}(p)$.

(69) For every finite partial state p of **SCM** holds $\mathrm{dom}\,\mathrm{DataPart}(p) \subseteq \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$.

(70) For every finite partial state p of **SCM** holds $\mathrm{dom}\,\mathrm{ProgramPart}(p) \subseteq \mathrm{Instr\text{-}Loc}_{\mathrm{SCM}}$.

(71) For all finite partial states p, q of S holds $\mathrm{dom}\,\mathrm{DataPart}(p)$ misses $\mathrm{dom}\,\mathrm{ProgramPart}(q)$.

(72) For every programmed finite partial state p of S holds $\mathrm{ProgramPart}(p) = p$.

(73) For every finite partial state p of S and for every instruction-location l of S such that $l \in \mathrm{dom}\,p$ holds $l \in \mathrm{dom}\,\mathrm{ProgramPart}(p)$.

(74) Let p be a data-only finite partial state of S and q be a finite partial state of S. Then $p \subseteq q$ if and only if $p \subseteq \mathrm{DataPart}(q)$.

(75) Let S be an IC-Ins-separated definite realistic non empty non void AMI over N and p be a finite partial state of S. If $\mathbf{IC}_S \in \mathrm{dom}\,p$, then $p = \mathrm{Start\text{-}At}(\mathbf{IC}_p) +\cdot \mathrm{ProgramPart}(p) +\cdot \mathrm{DataPart}(p)$.

Let us consider N, S and let I_1 be a partial function from $\mathrm{FinPartSt}(S)$ to $\mathrm{FinPartSt}(S)$. We say that I_1 is data-only if and only if the condition (Def. 9) is satisfied.

(Def. 9) Let p be a finite partial state of S. Suppose $p \in \mathrm{dom}\,I_1$. Then p is data-only and for every finite partial state q of S such that $q = I_1(p)$ holds q is data-only.

We now state the proposition

(76) Let S be an IC-Ins-separated definite realistic non empty non void AMI over N and p be a finite partial state of S. If $\mathbf{IC}_S \in \mathrm{dom}\,p$, then p is not programmed.

Let us consider N, let S be a non void AMI over N, let s be a state of S, and let p be a finite partial state of S. Then $s +\cdot p$ is a state of S.

We now state several propositions:

(77) Let i be an instruction of **SCM**, s be a state of **SCM**, and p be a programmed finite partial state of **SCM**. Then $\mathrm{Exec}(i, s +\cdot p) = \mathrm{Exec}(i, s) +\cdot p$.

(78) For every finite partial state p of S such that $\mathbf{IC}_S \in \mathrm{dom}\,p$ holds $\mathrm{Start\text{-}At}(\mathbf{IC}_p) \subseteq p$.

(79) For every state s of S and for every instruction-location i_3 of S holds $\mathbf{IC}_{s +\cdot \mathrm{Start\text{-}At}(i_3)} = i_3$.

(80) For every state s of **SCM** and for every instruction-location i_3 of **SCM** and for every data-location a holds $s(a) = (s +\cdot \mathrm{Start\text{-}At}(i_3))(a)$.

(81) Let S be an IC-Ins-separated definite realistic non empty non void AMI over N, s be a state of S, i_3 be an instruction-location of S, and a be an instruction-location of S. Then $s(a) = (s +\cdot \mathrm{Start\text{-}At}(i_3))(a)$.

(82) For all states s, t of S and for every set A holds $s +\cdot t \restriction A$ is a state of S.

4. AUTONOMIC FINITE PARTIAL STATES OF **SCM**

We now state the proposition

(83) For every autonomic finite partial state p of **SCM** such that $\mathrm{DataPart}(p) \neq \emptyset$ holds $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, p$.

Let us note that there exists a finite partial state of **SCM** which is autonomic and non programmed.

One can prove the following propositions:

(84) For every autonomic non programmed finite partial state p of **SCM** holds $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, p$.

(85) For every autonomic finite partial state p of **SCM** such that $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, p$ holds $\mathbf{IC}_{p} \in \mathrm{dom}\, p$.

(86) Let p be an autonomic non programmed finite partial state of **SCM** and s be a state of **SCM**. If $p \subseteq s$, then for every natural number i holds $\mathbf{IC}_{(\mathrm{Computation}(s))(i)} \in \mathrm{dom}\,\mathrm{ProgramPart}(p)$.

(87) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. If $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$, then $\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)} = \mathbf{IC}_{(\mathrm{Computation}(s_2))(i)}$ and $I = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(i))$.

(88) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. If $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$, then if $I = d_2{:=}d_3$ and $d_2 \in \mathrm{dom}\, p$, then $(\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_3)$.

(89) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \mathrm{AddTo}(d_2, d_3)$ and $d_2 \in \mathrm{dom}\, p$, then $(\mathrm{Computation}(s_1))(i)(d_2) + (\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_2) + (\mathrm{Computation}(s_2))(i)(d_3)$.

(90) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \mathrm{SubFrom}(d_2, d_3)$ and $d_2 \in \mathrm{dom}\, p$, then $(\mathrm{Computation}(s_1))(i)(d_2) - (\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_2) - (\mathrm{Computation}(s_2))(i)(d_3)$.

(91) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \mathrm{MultBy}(d_2, d_3)$ and $d_2 \in \mathrm{dom}\, p$, then $(\mathrm{Computation}(s_1))(i)(d_2) \cdot (\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_2) \cdot (\mathrm{Computation}(s_2))(i)(d_3)$.

(92) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \mathrm{Divide}(d_2, d_3)$ and $d_2 \in \mathrm{dom}\, p$ and $d_2 \neq d_3$, then $(\mathrm{Computation}(s_1))(i)(d_2) \div (\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_2) \div (\mathrm{Computation}(s_2))(i)(d_3)$.

(93) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \mathrm{Divide}(d_2, d_3)$ and $d_3 \in \mathrm{dom}\, p$ and $d_2 \neq d_3$, then $(\mathrm{Computation}(s_1))(i)(d_2) \bmod (\mathrm{Computation}(s_1))(i)(d_3) = (\mathrm{Computation}(s_2))(i)(d_2) \bmod (\mathrm{Computation}(s_2))(i)(d_3)$.

(94) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \textbf{if } d_2 = 0 \textbf{ goto } l_1$ and $l_1 \neq \mathrm{Next}(\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)})$, then $(\mathrm{Computation}(s_1))(i)(d_2) = 0$ iff $(\mathrm{Computation}(s_2))(i)(d_2) = 0$.

(95) Let p be an autonomic non programmed finite partial state of **SCM** and s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, d_2, d_3 be data-locations, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. Suppose $I = \mathrm{CurInstr}((\mathrm{Computation}(s_1))(i))$. If $I = \textbf{if } d_2 > 0 \textbf{ goto } l_1$ and $l_1 \neq \mathrm{Next}(\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)})$, then $(\mathrm{Computation}(s_1))(i)(d_2) > 0$ iff $(\mathrm{Computation}(s_2))(i)(d_2) > 0$.

(96) For every finite partial state p of **SCM** holds $\mathrm{DataPart}(p) = p \restriction \text{Data-Loc}_{\mathrm{SCM}}$.

(97) For every finite partial state f of **SCM** holds f is data-only iff $\mathrm{dom}\, f \subseteq \text{Data-Loc}_{\mathrm{SCM}}$.

REFERENCES

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

[2] Grzegorz Bancerek. The ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal1.html.

[3] Grzegorz Bancerek. Sequences of ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal2.html.

[4] Grzegorz Bancerek. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/card_3.html.

[5] 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.

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

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

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

[9] 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.

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

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

[12] 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.

[13] Takaya Nishiyama and Yasuho Mizuhara. Binary arithmetics. *Journal of Formalized Mathematics*, 5, 1993. http://mizar.org/JFM/Vol5/binarith.html.

[14] Dariusz Surowik. Cyclic groups and some of their properties — part I. *Journal of Formalized Mathematics*, 3, 1991. http://mizar.org/JFM/Vol3/gr_cy_1.html.

[15] Andrzej Trybulec. Domains and their Cartesian products. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/domain_1.html.

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

[17] Andrzej Trybulec. Tuples, projections and Cartesian products. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/mcart_1.html.

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

[19] 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`.

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

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

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

Received November 23, 1993

Published January 2, 2004