FORMALIZED MATHEMATICS
Volume 5, Number 1, 1996
Warsaw University - 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 [11,12,18]. 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.

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

1. Preliminaries

The following propositions are true:

(1) For all sets A, B, X, Y such that $A \subseteq X$ and $B \subseteq Y$ and $X \cap Y = \emptyset$ holds $A \cap B = \emptyset$.

(2) For all sets X, Y, Z such that $X \subseteq Y$ holds $X \subseteq Z \cup Y$ and $X \subseteq Y \cup Z$.

(3) 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 $\mathrm{dom}\, g$ holds $(f {+\cdot} g) \restriction A = f \restriction A$.

ISSN 0777–4028

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

(9) For all functions f, g, h such that $\mathrm{dom}\, g = \mathrm{dom}\, h$ 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 $\mathrm{dom}\, f \subseteq B$ and $\mathrm{dom}\, g \subseteq C$ 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 $\mathrm{dom}\, p \subseteq A$ and $\mathrm{dom}\, q$ misses A holds $(p +\cdot q) \restriction A = p$.

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

2. Total states of **SCM**

One can prove the following propositions:

(15) Let a be a data-location and let s be a state of **SCM**. Then $(\mathrm{Exec}(\mathrm{Divide}(a, a), s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ 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 arbitrary x such that $x \in \text{Data-Loc}_{\mathrm{SCM}}$ holds x is a data-location.

(17) For arbitrary x such that $x \in \text{Instr-Loc}_{\mathrm{SCM}}$ holds x is an instruction-location of **SCM**.

(18) 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 $d_1 \neq \mathbf{IC}_{\mathbf{SCM}}$.

(21) For every instruction-location i_1 of **SCM** holds $i_1 \neq \mathbf{IC}_{\mathbf{SCM}}$.

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

(23) The objects of $\mathbf{SCM} = \{\mathbf{IC}_{\mathbf{SCM}}\} \cup \text{Data-Loc}_{\mathrm{SCM}} \cup \text{Instr-Loc}_{\mathrm{SCM}}$.

(24) Let s be a state of **SCM**, and let d be a data-location, and let 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 **SCM** holds $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, s$.

(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 $\text{Data-Loc}_{\mathrm{SCM}} \subseteq \mathrm{dom}\, s$.

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

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

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

(31) $\text{Data-Loc}_{\rm SCM}$ is finite.

(32) The instruction locations of **SCM** is finite.

(33) $\text{Data-Loc}_{\rm SCM}$ misses $\text{Instr-Loc}_{\rm SCM}$.

(34) For every state s of **SCM** holds $\text{Start-At}(\mathbf{IC}_s) = s \restriction \{\mathbf{IC}_{\mathbf{SCM}}\}$.

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

Let I be an instruction of **SCM**. The functor $\text{InsCode}(I)$ yields a natural number and is defined as follows:

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

The functor ${}^{@}I$ yielding an element of $\text{Instr}_{\rm SCM}$ is defined by:

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

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

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

Let l_1 be an element of $\text{Data-Loc}_{\rm SCM}$. 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 $\text{InsCode}(l) \leq 8$.

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

One can prove the following propositions:

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

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

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

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

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

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

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

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

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

In the sequel d_2, d_3 denote data-locations and l_1 denotes an instruction-location of **SCM**.

We now state a number of propositions:

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

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

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

(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 $({}^{@}\mathrm{goto}\ l_1)\mathrm{address_j} = l_1$.

(56) For every instruction-location l_1 of **SCM** and for every data-location a holds $({}^{@}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1))\mathrm{address_j} = l_1$ and $({}^{@}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1))\mathrm{address_c} = a$.

(57) For every instruction-location l_1 of **SCM** and for every data-location a holds $({}^{@}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1))\mathrm{address_j} = l_1$ and $({}^{@}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1))\mathrm{address_c} = a$.

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

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

3. Finite partial states of SCM

The following proposition is true

(60) For every finite partial state p of **SCM** and for every state s of **SCM** such that $\mathbf{IC_{SCM}} \in \mathrm{dom}\, p$ and $p \subseteq s$ holds $\mathbf{IC}_p = \mathbf{IC}_s$.

Let p be a finite partial state of **SCM** and let l_1 be an instruction-location of **SCM**. Let us assume that $l_1 \in \mathrm{dom}\, p$. The functor $\pi_{l_1} p$ yielding an instruction of **SCM** is defined by:

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

The following proposition is true

(61) Let x be arbitrary and let p be a finite partial state of **SCM**. If $x \subseteq p$, then x is a finite partial state of **SCM**.

Let p be a finite partial state of **SCM**. The functor $\mathrm{ProgramPart}(p)$ yields a programmed finite partial state of **SCM** and is defined by:

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

The functor DataPart(p) yielding a finite partial state of **SCM** is defined as follows:

(Def.7) DataPart(p) = $p \upharpoonright$ Data-Loc$_{\mathrm{SCM}}$.

A finite partial state of **SCM** is data-only if:

(Def.8) dom it $\subseteq$ Data-Loc$_{\mathrm{SCM}}$.

Next we state a number of propositions:

(62) For every finite partial state p of **SCM** holds DataPart(p) $\subseteq p$.

(63) For every finite partial state p of **SCM** holds ProgramPart(p) $\subseteq p$.

(64) Let p be a finite partial state of **SCM** and let s be a state of **SCM**. If $p \subseteq s$, then for every natural number i holds ProgramPart(p) $\subseteq$ (Computation(s))(i).

(65) For every finite partial state p of **SCM** holds $\mathbf{IC}_{\mathbf{SCM}} \notin$ dom DataPart(p).

(66) For every finite partial state p of **SCM** holds $\mathbf{IC}_{\mathbf{SCM}} \notin$ dom ProgramPart(p).

(67) For every finite partial state p of **SCM** holds $\{\mathbf{IC}_{\mathbf{SCM}}\}$ misses dom DataPart(p).

(68) For every finite partial state p of **SCM** holds $\{\mathbf{IC}_{\mathbf{SCM}}\}$ misses dom ProgramPart(p).

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

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

(71) For all finite partial states p, q of **SCM** holds dom DataPart(p) misses dom ProgramPart(q).

(72) For every programmed finite partial state p of **SCM** holds ProgramPart(p) = p.

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

(74) Let p be a data-only finite partial state of **SCM** and let q be a finite partial state of **SCM**. Then $p \subseteq q$ if and only if $p \subseteq$ DataPart(q).

(75) For every finite partial state p of **SCM** such that $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$ holds p = Start-At($\mathbf{IC}_p$) +· ProgramPart(p) +· DataPart(p).

A partial function from FinPartSt(**SCM**) to FinPartSt(**SCM**) is data-only if it satisfies the condition (Def.9).

(Def.9) Let p be a finite partial state of **SCM**. Suppose $p \in$ dom it. Then p is data-only and for every finite partial state q of **SCM** such that q = it(p) holds q is data-only.

Next we state the proposition

(76) For every finite partial state p of **SCM** such that $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$ holds p is not programmed.

Let s be a state of **SCM** and let p be a finite partial state of **SCM**. Then $s +\cdot p$ is a state of **SCM**.

Next we state several propositions:

(77) Let i be an instruction of **SCM**, and let s be a state of **SCM**, and let 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 **SCM** such that $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, p$ holds $\text{Start-At}(\mathbf{IC}_p) \subseteq p$.

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

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

(81) Let s be a state of **SCM**, and let i_3 be an instruction-location of **SCM**, and let a be an instruction-location of **SCM**. Then $s(a) = (s +\cdot \text{Start-At}(i_3))(a)$.

(82) For all states s, t of **SCM** holds $s +\cdot t \restriction \text{Data-Loc}_{\text{SCM}}$ is a state of **SCM**.

4. Autonomic finite partial states of SCM

The following proposition is true

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

One can check that there exists a finite partial state of **SCM** which is autonomic and non programmed.

We now state a number of 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 let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an

instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 let s_1, s_2 be states of **SCM**. Suppose $p \subseteq s_1$ and $p \subseteq s_2$. Let i be a natural number, and let d_2, d_3 be data-locations, and let l_1 be an instruction-location of **SCM**, and let 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 $(\mathrm{Computation}(s_2))(i)(d_2) > 0$.

References

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
[2] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.
[3] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[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. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[7] 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.
[8] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.
[9] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.
[10] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[11] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[12] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[13] Takaya Nishiyama and Yasuho Mizuhara. Binary arithmetics. *Formalized Mathematics*, 4(**1**):83–86, 1993.
[14] Dariusz Surowik. Cyclic groups and some of their properties - part I. *Formalized Mathematics*, 2(**5**):623–627, 1991.
[15] Andrzej Trybulec. Domains and their Cartesian products. *Formalized Mathematics*, 1(**1**):115–122, 1990.
[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] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[20] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received November 23, 1993