FORMALIZED MATHEMATICS
Volume 5, Number 1, 1996
Warsaw University - Białystok

Relocatability [1]

Yasushi Tanaka
Shinshu University
Information Engineering Dept.
Nagano

Summary. This article defines the concept of relocating the program part of a finite partial state of **SCM** (data part stays intact). The relocated program differs from the original program in that all jump instructions are adjusted by the relocation factor and other instructions remain unchanged. The main theorem states that if a program computes a function then the relocated program computes the same function, and vice versa.

MML Identifier: RELOC.

The terminology and notation used in this paper have been introduced in the following articles: [16], [2], [1], [19], [5], [6], [15], [7], [18], [13], [4], [9], [3], [8], [10], [11], [17], [12], and [14].

1. Relocatability

In this paper j, k, m will be natural numbers.

Let l_1 be an instruction-location of **SCM** and let k be a natural number. The functor $l_1 + k$ yielding an instruction-location of **SCM** is defined as follows:

(Def.1) There exists a natural number m such that $l_1 = \mathbf{i}_m$ and $l_1 + k = \mathbf{i}_{m+k}$.

The functor $l_1 -' k$ yields an instruction-location of **SCM** and is defined as follows:

(Def.2) There exists a natural number m such that $l_1 = \mathbf{i}_m$ and $l_1 -' k = \mathbf{i}_{m-'k}$.

The following three propositions are true:

(1) For every instruction-location l_1 of **SCM** and for every natural number k holds $(l_1 + k) -' k = l_1$.

[1] This work was done under guidance and supervision of A. Trybulec and P. Rudnicki.

ISSN 0777–4028

(2) For all instructions-locations l_2, l_3 of **SCM** and for every natural number k holds Start-At$(l_2 + k)$ = Start-At$(l_3 + k)$ iff Start-At(l_2) = Start-At(l_3).

(3) For all instructions-locations l_2, l_3 of **SCM** and for every natural number k such that Start-At(l_2) = Start-At(l_3) holds Start-At$(l_2 -' k)$ = Start-At$(l_3 -' k)$.

Let I be an instruction of **SCM** and let k be a natural number. The functor IncAddr(I, k) yields an instruction of **SCM** and is defined as follows:

(Def.3) (i) IncAddr(I, k) = goto $((^{@}I)\text{address}_{\text{j}}^{\text{T}} + k)$ if InsCode$(I) = 6$,

(ii) IncAddr(I, k) = **if** $(^{@}I)\text{address}_{\text{c}}^{\text{T}} = 0$ **goto** $(^{@}I)\text{address}_{\text{j}}^{\text{T}} + k$ if InsCode$(I) = 7$,

(iii) IncAddr(I, k) = **if** $(^{@}I)\text{address}_{\text{c}}^{\text{T}} > 0$ **goto** $(^{@}I)\text{address}_{\text{j}}^{\text{T}} + k$ if InsCode$(I) = 8$,

(iv) IncAddr$(I, k) = I$, otherwise.

One can prove the following propositions:

(4) For every natural number k holds IncAddr$(\mathbf{halt}_{\mathbf{SCM}}, k) = \mathbf{halt}_{\mathbf{SCM}}$.

(5) For every natural number k and for all data-locations a, b holds IncAddr$(a{:=}b, k) = a{:=}b$.

(6) For every natural number k and for all data-locations a, b holds IncAddr(AddTo$(a, b), k)$ = AddTo(a, b).

(7) For every natural number k and for all data-locations a, b holds IncAddr(SubFrom$(a, b), k)$ = SubFrom(a, b).

(8) For every natural number k and for all data-locations a, b holds IncAddr(MultBy$(a, b), k)$ = MultBy(a, b).

(9) For every natural number k and for all data-locations a, b holds IncAddr(Divide$(a, b), k)$ = Divide(a, b).

(10) For every natural number k and for every instruction-location l_1 of **SCM** holds IncAddr(goto $l_1, k)$ = goto $(l_1 + k)$.

(11) Let k be a natural number, and let l_1 be an instruction-location of **SCM**, and let a be a data-location. Then IncAddr(**if** $a = 0$ **goto** $l_1, k)$ = **if** $a = 0$ **goto** $l_1 + k$.

(12) Let k be a natural number, and let l_1 be an instruction-location of **SCM**, and let a be a data-location. Then IncAddr(**if** $a > 0$ **goto** $l_1, k)$ = **if** $a > 0$ **goto** $l_1 + k$.

(13) For every instruction I of **SCM** and for every natural number k holds InsCode(IncAddr$(I, k))$ = InsCode(I).

(14) Let I_1, I be instructions of **SCM** and let k be a natural number. Suppose InsCode$(I) = 0$ or InsCode$(I) = 1$ or InsCode$(I) = 2$ or InsCode$(I) = 3$ or InsCode$(I) = 4$ or InsCode$(I) = 5$ but IncAddr$(I_1, k) = I$. Then $I_1 = I$.

Let p be a programmed finite partial state of **SCM** and let k be a natural number. The functor Shift(p, k) yielding a programmed finite partial state of

SCM is defined by:

(Def.4) $\operatorname{dom}\operatorname{Shift}(p,k) = \{\mathbf{i}_{m+k} : \mathbf{i}_m \in \operatorname{dom} p\}$ and for every m such that $\mathbf{i}_m \in \operatorname{dom} p$ holds $(\operatorname{Shift}(p,k))(\mathbf{i}_{m+k}) = p(\mathbf{i}_m)$.

We now state three propositions:

(15) Let l be an instruction-location of **SCM**, and let k be a natural number, and let p be a programmed finite partial state of **SCM**. If $l \in \operatorname{dom} p$, then $(\operatorname{Shift}(p,k))(l+k) = p(l)$.

(16) Let p be a programmed finite partial state of **SCM** and let k be a natural number. Then $\operatorname{dom}\operatorname{Shift}(p,k) = \{i_1 + k : i_1$ ranges over instructions-locations of **SCM**, $i_1 \in \operatorname{dom} p\}$.

(17) Let p be a programmed finite partial state of **SCM** and let k be a natural number. Then $\operatorname{dom}\operatorname{Shift}(p,k) \subseteq$ the instruction locations of **SCM**.

Let p be a programmed finite partial state of **SCM** and let k be a natural number. The functor $\operatorname{IncAddr}(p,k)$ yielding a programmed finite partial state of **SCM** is defined as follows:

(Def.5) $\operatorname{dom}\operatorname{IncAddr}(p,k) = \operatorname{dom} p$ and for every m such that $\mathbf{i}_m \in \operatorname{dom} p$ holds $(\operatorname{IncAddr}(p,k))(\mathbf{i}_m) = \operatorname{IncAddr}(\pi_{\mathbf{i}_m} p, k)$.

One can prove the following two propositions:

(18) Let p be a programmed finite partial state of **SCM**, and let k be a natural number, and let l be an instruction-location of **SCM**. If $l \in \operatorname{dom} p$, then $(\operatorname{IncAddr}(p,k))(l) = \operatorname{IncAddr}(\pi_l p, k)$.

(19) For every natural number i and for every programmed finite partial state p of **SCM** holds $\operatorname{Shift}(\operatorname{IncAddr}(p,i),i) = \operatorname{IncAddr}(\operatorname{Shift}(p,i),i)$.

Let p be a finite partial state of **SCM** and let k be a natural number. The functor $\operatorname{Relocated}(p,k)$ yielding a finite partial state of **SCM** is defined as follows:

(Def.6) $\operatorname{Relocated}(p,k) = \text{Start-At}(\mathbf{IC}_p + k) +\cdot \operatorname{IncAddr}(\operatorname{Shift}(\operatorname{ProgramPart}(p), k), k) +\cdot \operatorname{DataPart}(p)$.

Next we state a number of propositions:

(20) For every finite partial state p of **SCM** holds $\operatorname{dom}\operatorname{IncAddr}(\operatorname{Shift}(\operatorname{ProgramPart}(p),k),k) \subseteq \text{Instr-Loc}_{\text{SCM}}$.

(21) For every finite partial state p of **SCM** and for every natural number k holds $\operatorname{DataPart}(\operatorname{Relocated}(p,k)) = \operatorname{DataPart}(p)$.

(22) For every finite partial state p of **SCM** and for every natural number k holds $\operatorname{ProgramPart}(\operatorname{Relocated}(p,k)) = \operatorname{IncAddr}(\operatorname{Shift}(\operatorname{ProgramPart}(p),k),k)$.

(23) For every finite partial state p of **SCM** holds $\operatorname{dom}\operatorname{ProgramPart}(\operatorname{Relocated}(p,k)) = \{\mathbf{i}_{j+k} : \mathbf{i}_j \in \operatorname{dom}\operatorname{ProgramPart}(p)\}$.

(24) Let p be a finite partial state of **SCM**, and let k be a natural number, and let l be an instruction-location of **SCM**. Then $l \in \operatorname{dom} p$ if and only if $l + k \in \operatorname{dom}\operatorname{Relocated}(p,k)$.

(25) For every finite partial state p of **SCM** and for every natural number k holds $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom}\operatorname{Relocated}(p,k)$.

(26) For every finite partial state p of **SCM** and for every natural number k holds $\mathbf{IC}_{\mathrm{Relocated}(p,k)} = \mathbf{IC}_p + k$.

(27) Let p be a finite partial state of **SCM**, and let k be a natural number, and let l_1 be an instruction-location of **SCM**, and let I be an instruction of **SCM**. If $l_1 \in \operatorname{dom} \mathrm{ProgramPart}(p)$ and $I = p(l_1)$, then $\mathrm{IncAddr}(I, k) = (\mathrm{Relocated}(p, k))(l_1 + k)$.

(28) For every finite partial state p of **SCM** and for every natural number k holds $\text{Start-At}(\mathbf{IC}_p + k) \subseteq \mathrm{Relocated}(p, k)$.

(29) Let s be a data-only finite partial state of **SCM**, and let p be a finite partial state of **SCM**, and let k be a natural number. If $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$, then $\mathrm{Relocated}(p +\cdot s, k) = \mathrm{Relocated}(p, k) +\cdot s$.

(30) Let k be a natural number, and let p be an autonomic finite partial state of **SCM**, and let s_1, s_2 be states of **SCM**. If $p \subseteq s_1$ and $\mathrm{Relocated}(p, k) \subseteq s_2$, then $p \subseteq s_1 +\cdot s_2 \restriction \text{Data-Loc}_{\mathrm{SCM}}$.

(31) For every state s of **SCM** holds $\mathrm{Exec}(\mathrm{IncAddr}(\mathrm{CurInstr}(s), k), s +\cdot \text{Start-At}(\mathbf{IC}_s + k)) = \mathrm{Following}(s) +\cdot \text{Start-At}(\mathbf{IC}_{\mathrm{Following}(s)} + k)$.

(32) Let I_2 be an instruction of **SCM**, and let s be a state of **SCM**, and let p be a finite partial state of **SCM**, and let i, j, k be natural numbers. If $\mathbf{IC}_s = \mathbf{i}_{j+k}$, then $\mathrm{Exec}(I_2, s +\cdot \text{Start-At}(\mathbf{IC}_s -' k)) = \mathrm{Exec}(\mathrm{IncAddr}(I_2, k), s) +\cdot \text{Start-At}(\mathbf{IC}_{\mathrm{Exec}(\mathrm{IncAddr}(I_2,k),s)} -' k)$.

2. Main theorems of Relocatability

Next we state several propositions:

(33) Let k be a natural number and let p be an autonomic finite partial state of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$. Let s be a state of **SCM**. Suppose $p \subseteq s$. Let i be a natural number. Then $(\mathrm{Computation}(s +\cdot \mathrm{Relocated}(p, k)))(i) = (\mathrm{Computation}(s))(i) +\cdot \text{Start-At}(\mathbf{IC}_{(\mathrm{Computation}(s))(i)} + k) +\cdot \mathrm{ProgramPart}(\mathrm{Relocated}(p, k))$.

(34) Let k be a natural number, and let p be an autonomic finite partial state of **SCM**, and let s_1, s_2, s_3 be states of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$ and $p \subseteq s_1$ and $\mathrm{Relocated}(p, k) \subseteq s_2$ and $s_3 = s_1 +\cdot s_2 \restriction \text{Data-Loc}_{\mathrm{SCM}}$. Let i be a natural number. Then $\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)} + k = \mathbf{IC}_{(\mathrm{Computation}(s_2))(i)}$ and $\mathrm{IncAddr}(\mathrm{CurInstr}((\mathrm{Computation}(s_1))(i)), k) = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(i))$ and $(\mathrm{Computation}(s_1))(i) \restriction \operatorname{dom} \mathrm{DataPart}(p) = (\mathrm{Computation}(s_2))(i) \restriction \operatorname{dom} \mathrm{DataPart}(\mathrm{Relocated}(p, k))$ and $(\mathrm{Computation}(s_3))(i) \restriction \text{Data-Loc}_{\mathrm{SCM}} = (\mathrm{Computation}(s_2))(i) \restriction \text{Data-Loc}_{\mathrm{SCM}}$.

(35) Let p be an autonomic finite partial state of **SCM** and let k be a natural number. If $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$, then p is halting iff $\mathrm{Relocated}(p, k)$ is halting.

(36) Let k be a natural number and let p be an autonomic finite partial state of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$. Let s be a

state of **SCM**. Suppose Relocated$(p, k) \subseteq s$. Let i be a natural number. Then (Computation$(s))(i)$ = (Computation$(s +\cdot p))(i) +\cdot$ Start-At$(\mathbf{IC}_{(\text{Computation}(s+\cdot p))(i)} + k) +\cdot s \restriction \operatorname{dom}$ ProgramPart$(p) +\cdot$ ProgramPart(Relocated(p, k)).

(37) Let k be a natural number and let p be a finite partial state of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$. Let s be a state of **SCM**. Suppose $p \subseteq s$ and Relocated(p, k) is autonomic. Let i be a natural number. Then (Computation$(s))(i)$ = (Computation$(s +\cdot$ Relocated$(p, k)))(i) +\cdot$ Start-At$(\mathbf{IC}_{(\text{Computation}(s+\cdot\text{Relocated}(p,k)))(i)} -' k) +\cdot$ $s \restriction \operatorname{dom}$ ProgramPart(Relocated$(p, k)) +\cdot$ ProgramPart(p).

(38) Let p be a finite partial state of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$. Let k be a natural number. Then p is autonomic if and only if Relocated(p, k) is autonomic.

(39) Let p be a halting autonomic finite partial state of **SCM**. If $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$, then for every natural number k holds DataPart(Result$(p))$ = DataPart(Result(Relocated$(p, k))$).

(40) Let F be a data-only partial function from FinPartSt(**SCM**) to FinPartSt(**SCM**) and let p be a finite partial state of **SCM**. Suppose $\mathbf{IC}_{\mathbf{SCM}} \in \operatorname{dom} p$. Let k be a natural number. Then p computes F if and only if Relocated(p, k) computes F.

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] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[11] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[12] Takaya Nishiyama and Yasuho Mizuhara. Binary arithmetics. *Formalized Mathematics*, 4(**1**):83–86, 1993.
[13] Dariusz Surowik. Cyclic groups and some of their properties - part I. *Formalized Mathematics*, 2(**5**):623–627, 1991.
[14] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[15] Andrzej Trybulec. Binary operations applied to functions. *Formalized Mathematics*, 1(**2**):329–334, 1990.

[16] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[17] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[18] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[19] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received June 16, 1994