JOURNAL OF FORMALIZED MATHEMATICS
Volume 6, Released 1994, Published 2003
Inst. of Computer Science, Univ. of 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`.

WWW: `http://mizar.org/JFM/Vol6/reloc.html`

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

1. Relocatability

In this paper j, k, m are 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 by:

(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 $l_1 -' k$ yielding an instruction-location of **SCM** is defined by:

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

Next we state three propositions:

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

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

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

Let I be an instruction of **SCM** and let k be a natural number. The functor $\text{IncAddr}(I, k)$ yielding an instruction of **SCM** is defined by:

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

(Def. 3) $\mathrm{IncAddr}(I,k) = \begin{cases} \mathrm{goto}\ ((^{@}I)\,\mathrm{address}_{\mathrm{j}}^{\mathrm{T}} + k), & \text{if } \mathrm{InsCode}(I) = 6, \\ \mathbf{if}\ (^{@}I)\,\mathrm{address}_{\mathrm{c}}^{\mathrm{T}} = 0\ \mathbf{goto}\ (^{@}I)\,\mathrm{address}_{\mathrm{j}}^{\mathrm{T}} + k, & \text{if } \mathrm{InsCode}(I) = 7, \\ \mathbf{if}\ (^{@}I)\,\mathrm{address}_{\mathrm{c}}^{\mathrm{T}} > 0\ \mathbf{goto}\ (^{@}I)\,\mathrm{address}_{\mathrm{j}}^{\mathrm{T}} + k, & \text{if } \mathrm{InsCode}(I) = 8, \\ I, & \text{otherwise.} \end{cases}$

One can prove the following propositions:

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

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

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

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

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

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

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

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

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

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

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

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

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

Next we state three propositions:

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

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

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

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

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

Next we state two propositions:

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

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

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

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

Next we state a number of propositions:

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

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

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

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

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

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

(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**, k be a natural number, l_1 be an instruction-location of **SCM**, and I be an instruction of **SCM**. If $l_1 \in \mathrm{dom}\,\mathrm{ProgramPart}(p)$ 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**, p be a finite partial state of **SCM**, and k be a natural number. If $\mathbf{IC}_{\mathbf{SCM}} \in \mathrm{dom}\, p$, then $\mathrm{Relocated}(p +\cdot s, k) = \mathrm{Relocated}(p,k) +\cdot s$.

(30) Let k be a natural number, p be an autonomic finite partial state of **SCM**, and 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**, s be a state of **SCM**, p be a finite partial state of **SCM**, and 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 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 $(\text{Computation}(s{+}\cdot\text{Relocated}(p,k)))(i) = (\text{Computation}(s))(i){+}\cdot\text{Start-At}(\mathbf{IC}_{(\text{Computation}(s))(i)} + k){+}\cdot\text{ProgramPart}(\text{Relocated}(p,k))$.

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

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

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

(37) Let k be a natural number and 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 $\text{Relocated}(p,k)$ is autonomic. Let i be a natural number. Then $(\text{Computation}(s))(i) = (\text{Computation}(s{+}\cdot\text{Relocated}(p,k)))(i){+}\cdot\text{Start-At}(\mathbf{IC}_{(\text{Computation}(s+\cdot\text{Relocated}(p,k)))(i)} - k){+}\cdot s{\restriction}\operatorname{dom}\text{ProgramPart}(\text{Relocated}(p,k)){+}\cdot\text{ProgramPart}(p)$.

(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 $\text{Relocated}(p,k)$ 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 $\text{DataPart}(\text{Result}(p)) = \text{DataPart}(\text{Result}(\text{Relocated}(p,k)))$.

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

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

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

[4] Czesław Byliński. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funct_2.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] Agata Darmochwał. Finite sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finset_1.html.

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

[9] 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.

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

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

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

[13] 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`.

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

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

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

Received June 16, 1994

Published January 2, 2004