FORMALIZED MATHEMATICS
Volume 5, Number 4, 1996
Warsaw University - Białystok

Modifying Addresses of Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$

Andrzej Trybulec
Warsaw University
Białystok

Yatsuka Nakamura
Shinshu University
Nagano

MML Identifier: `SCMFSA_4`.

The notation and terminology used in this paper are introduced in the following papers: [10], [1], [13], [14], [21], [18], [23], [17], [24], [6], [7], [8], [4], [3], [2], [9], [5], [22], [11], [12], [19], [15], [16], and [20].

1. Preliminaries

Let N be a non empty set with non empty elements and let S be an AMI over N. One can check that every finite partial state of S is finite.

Let N be a non empty set with non empty elements and let S be an AMI over N. One can verify that there exists a finite partial state of S which is programmed.

Next we state the proposition

(1) Let N be a non empty set with non empty elements, and let S be a definite AMI over N, and let p be a programmed finite partial state of S. Then $\operatorname{rng} p \subseteq$ the instructions of S.

Let N be a non empty set with non empty elements, let S be a definite AMI over N, and let I, J be programmed finite partial states of S. Then $I {+}\!\cdot J$ is a programmed finite partial state of S.

Next we state the proposition

(2) Let N be a non empty set with non empty elements, and let S be a definite AMI over N, and let f be a function from the instructions of S into the instructions of S, and let s be a programmed finite partial state of S. Then $\operatorname{dom}(f \cdot s) = \operatorname{dom} s$.

ISSN 1426–2630

2. Incrementing and decrementing the instruction locations

In the sequel i, k, l, m, n, p will denote natural numbers.

Let l_1 be an instruction-location of $\mathbf{SCM}_{\rm FSA}$ and let k be a natural number. The functor $l_1 + k$ yielding an instruction-location of $\mathbf{SCM}_{\rm FSA}$ is defined by:

(Def. 1) There exists a natural number m such that $l_1 = \mathrm{insloc}(m)$ and $l_1 + k = \mathrm{insloc}(m + k)$.

The functor $l_1 -' k$ yields an instruction-location of $\mathbf{SCM}_{\rm FSA}$ and is defined by:

(Def. 2) There exists a natural number m such that $l_1 = \mathrm{insloc}(m)$ and $l_1 -' k = \mathrm{insloc}(m -' k)$.

We now state two propositions:

(3) For every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ and for all m, n holds $(l + m) + n = l + (m + n)$.

(4) For every instruction-location l_1 of $\mathbf{SCM}_{\rm FSA}$ and for every natural number k holds $(l_1 + k) -' k = l_1$.

In the sequel L will be an instruction-location of $\mathbf{SCM}$ and I will be an instruction of $\mathbf{SCM}$.

The following three propositions are true:

(5) For every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ and for every L such that $L = l$ holds $l + k = L + k$.

(6) For all instructions-locations l_2, l_3 of $\mathbf{SCM}_{\rm FSA}$ and for every natural number k holds $\text{Start-At}(l_2 + k) = \text{Start-At}(l_3 + k)$ iff $\text{Start-At}(l_2) = \text{Start-At}(l_3)$.

(7) For all instructions-locations l_2, l_3 of $\mathbf{SCM}_{\rm FSA}$ and for every natural number k such that $\text{Start-At}(l_2) = \text{Start-At}(l_3)$ holds $\text{Start-At}(l_2 -' k) = \text{Start-At}(l_3 -' k)$.

3. Incrementing addresses

Let i be an instruction of $\mathbf{SCM}_{\rm FSA}$ and let k be a natural number. The functor $\mathrm{IncAddr}(i, k)$ yielding an instruction of $\mathbf{SCM}_{\rm FSA}$ is defined as follows:

(Def. 3) (i) There exists an instruction I of $\mathbf{SCM}$ such that $I = i$ and $\mathrm{IncAddr}(i, k) = \mathrm{IncAddr}(I, k)$ if $\mathrm{InsCode}(i) \in \{6, 7, 8\}$,

(ii) $\mathrm{IncAddr}(i, k) = i$, otherwise.

We now state a number of propositions:

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

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

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

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

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

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

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

(15) Let k be a natural number, and let l_1 be an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$, and let a be an integer location. Then $\mathrm{IncAddr}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1,k) = \mathbf{if}\ a=0\ \mathbf{goto}\ l_1+k$.

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

(17) Let k be a natural number, and let a, b be integer locations, and let f be a finite sequence location. Then $\mathrm{IncAddr}(b{:=}f_a,k) = b{:=}f_a$.

(18) Let k be a natural number, and let a, b be integer locations, and let f be a finite sequence location. Then $\mathrm{IncAddr}(f_a{:=}b,k) = f_a{:=}b$.

(19) Let k be a natural number, and let a be an integer location, and let f be a finite sequence location. Then $\mathrm{IncAddr}(a{:=}\mathrm{len}f,k) = a{:=}\mathrm{len}f$.

(20) Let k be a natural number, and let a be an integer location, and let f be a finite sequence location. Then $\mathrm{IncAddr}(f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle,k) = f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$.

(21) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every I such that $i = I$ holds $\mathrm{IncAddr}(i,k) = \mathrm{IncAddr}(I,k)$.

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

Let I_1 be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. We say that I_1 is initial if and only if:

(Def. 4) For all m, n such that $\mathrm{insloc}(n) \in \mathrm{dom}\, I_1$ and $m<n$ holds $\mathrm{insloc}(m) \in \mathrm{dom}\, I_1$.

The finite partial state $\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}$ of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined as follows:

(Def. 5) $\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}} = \mathrm{insloc}(0) \mapsto \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

Let us note that $\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}$ is non empty initial and programmed.

One can verify that there exists a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$ which is initial programmed and non empty.

Let f be a function and let g be a finite function. Note that $f \cdot g$ is finite.

Let N be a non empty set with non empty elements, let S be a definite AMI over N, let s be a programmed finite partial state of S, and let f be a function from the instructions of S into the instructions of S. Then $f \cdot s$ is a programmed finite partial state of S.

In the sequel i will denote an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$.

The following proposition is true

(23) $\mathrm{IncAddr}(\mathrm{IncAddr}(i, m), n) = \mathrm{IncAddr}(i, m + n)$.

4. Incremeting addresses in a finite partial state

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

(Def. 6) $\mathrm{dom}\,\mathrm{IncAddr}(p, k) = \mathrm{dom}\,p$ and for every m such that $\mathrm{insloc}(m) \in \mathrm{dom}\,p$ holds $(\mathrm{IncAddr}(p, k))(\mathrm{insloc}(m)) = \mathrm{IncAddr}(\pi_{\mathrm{insloc}(m)} p, k)$.

The following propositions are true:

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

(25) For all programmed finite partial states I, J of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{IncAddr}(I {+\cdot} J, n) = \mathrm{IncAddr}(I, n) {+\cdot} \mathrm{IncAddr}(J, n)$.

(26) Let f be a function from the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ into the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $f = \mathrm{id}_{(\text{the instructions of } \mathbf{SCM}_{\mathrm{FSA}})} {+\cdot} (\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}} \dot{\longmapsto} i)$. Let s be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $\mathrm{IncAddr}(f \cdot s, n) = (\mathrm{id}_{(\text{the instructions of } \mathbf{SCM}_{\mathrm{FSA}})} {+\cdot} (\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}} \dot{\longmapsto} \mathrm{IncAddr}(i, n))) \cdot \mathrm{IncAddr}(s, n)$.

(27) For every programmed finite partial state I of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{IncAddr}(\mathrm{IncAddr}(I, m), n) = \mathrm{IncAddr}(I, m + n)$.

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

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

5. Shifting the finite partial state

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

(Def. 7) $\operatorname{dom}\operatorname{Shift}(p,k) = \{\operatorname{insloc}(m+k) : \operatorname{insloc}(m) \in \operatorname{dom} p\}$ and for every m such that $\operatorname{insloc}(m) \in \operatorname{dom} p$ holds $(\operatorname{Shift}(p,k))(\operatorname{insloc}(m+k)) = p(\operatorname{insloc}(m))$.

The following propositions are true:

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

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

(32) For every programmed finite partial state I of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\operatorname{Shift}(\operatorname{Shift}(I,m),n) = \operatorname{Shift}(I,m+n)$.

(33) Let s be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$, and let f be a function from the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ into the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$, and given n. Then $\operatorname{Shift}(f \cdot s, n) = f \cdot \operatorname{Shift}(s,n)$.

(34) For all programmed finite partial states I, J of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\operatorname{Shift}(I {+\cdot} J, n) = \operatorname{Shift}(I,n) {+\cdot} \operatorname{Shift}(J,n)$.

(35) For all natural numbers i, j and for every programmed finite partial state p of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\operatorname{Shift}(\operatorname{IncAddr}(p,i),j) = \operatorname{IncAddr}(\operatorname{Shift}(p,j),i)$.

References

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.

[2] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.

[3] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.

[4] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.

[5] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.

[6] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.

[7] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.

[8] 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.

[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] Jan Popiołek. Some properties of functions modul and signum. *Formalized Mathematics*, 1(**2**):263–264, 1990.

[15] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.

[16] Yasushi Tanaka. Relocatability. *Formalized Mathematics*, 5(**1**):103–108, 1996.

[17] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.

[18] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.

[19] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.

[20] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\rm FSA}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.

[21] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.

[22] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.

[23] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.

[24] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received February 14, 1996