JOURNAL OF FORMALIZED MATHEMATICS
Volume 8, Released 1996, Published 2003
Inst. of Computer Science, Univ. of Białystok

On the Composition of Macro Instructions. Part III[1]

Noriko Asamoto
Ochanomizu University
Tokyo

Yatsuka Nakamura
Shinshu University
Nagano

Piotr Rudnicki
University of Alberta
Edmonton

Andrzej Trybulec
Warsaw University
Białystok

Summary. This article is a continuation of [15] and [2]. First, we recast the semantics of the macro composition in more convenient terms. Then, we introduce terminology and basic properties of macros constructed out of single instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. We give the complete semantics of composing a macro instruction with an instruction and for composing two machine instructions (this is also done in terms of macros). The introduced terminology is tested on the simple example of a macro for swapping two integer locations.

MML Identifier: `SCMFSA6C`.

WWW: `http://mizar.org/JFM/Vol8/scmfsa6c.html`

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

1. Preliminaries

For simplicity, we follow the rules: i is an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, a, b are integer locations, f is a finite sequence location, l is an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$, and s, s_1, s_2 are states of $\mathbf{SCM}_{\mathrm{FSA}}$.

The following two propositions are true:

(1) Let I be a keeping 0 parahalting macro instruction and J be a parahalting macro instruction. Then $(\mathrm{IExec}(I;J,s))(a) = (\mathrm{IExec}(J,\mathrm{IExec}(I,s)))(a)$.

(2) Let I be a keeping 0 parahalting macro instruction and J be a parahalting macro instruction. Then $(\mathrm{IExec}(I;J,s))(f) = (\mathrm{IExec}(J,\mathrm{IExec}(I,s)))(f)$.

2. Parahalting and keeping 0 macro instructions

Let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. We say that i is parahalting if and only if:

(Def. 1) $\mathrm{Macro}(i)$ is parahalting.

We say that i is keeping 0 if and only if:

(Def. 2) $\mathrm{Macro}(i)$ is keeping 0.

[1]This work was partially supported by NSERC Grant OGP9207 and NATO CRG 951368.

Let us observe that $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ is keeping 0 and parahalting.
Let us observe that there exists an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ which is keeping 0 and parahalting.
Let i be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that $\mathrm{Macro}(i)$ is parahalting.
Let i be a keeping 0 instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Note that $\mathrm{Macro}(i)$ is keeping 0.
Let a, b be integer locations. One can check the following observations:

* $a{:=}b$ is parahalting,
* $\mathrm{AddTo}(a,b)$ is parahalting,
* $\mathrm{SubFrom}(a,b)$ is parahalting,
* $\mathrm{MultBy}(a,b)$ is parahalting, and
* $\mathrm{Divide}(a,b)$ is parahalting.

Let f be a finite sequence location. Observe that $b{:=}f_a$ is parahalting and $f_a{:=}b$ is parahalting and keeping 0.

Let a be an integer location and let f be a finite sequence location. One can check that $a{:=}\mathrm{len}f$ is parahalting and $f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$ is parahalting and keeping 0.

Let a be a read-write integer location and let b be an integer location. One can check the following observations:

* $a{:=}b$ is keeping 0,
* $\mathrm{AddTo}(a,b)$ is keeping 0,
* $\mathrm{SubFrom}(a,b)$ is keeping 0, and
* $\mathrm{MultBy}(a,b)$ is keeping 0.

Let a, b be read-write integer locations. Observe that $\mathrm{Divide}(a,b)$ is keeping 0.

Let a be an integer location, let f be a finite sequence location, and let b be a read-write integer location. One can check that $b{:=}f_a$ is keeping 0.

Let f be a finite sequence location and let b be a read-write integer location. Note that $b{:=}\mathrm{len}f$ is keeping 0.

Let i be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let J be a parahalting macro instruction. Observe that $i;J$ is parahalting.

Let I be a parahalting macro instruction and let j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that $I;j$ is parahalting.

Let i be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Note that $i;j$ is parahalting.

Let i be a keeping 0 instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let J be a keeping 0 macro instruction. One can check that $i;J$ is keeping 0.

Let I be a keeping 0 macro instruction and let j be a keeping 0 instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that $I;j$ is keeping 0.

Let i, j be keeping 0 instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. One can verify that $i;j$ is keeping 0.

3. Semantics of compositions

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{Initialize}(s)$ yields a state of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 3) $\mathrm{Initialize}(s) = s{+\cdot}(\mathrm{intloc}(0) \longmapsto 1){+\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0))$.

The following propositions are true:

(3)(i) $\mathbf{IC}_{\mathrm{Initialize}(s)} = \mathrm{insloc}(0)$,

(ii) $(\mathrm{Initialize}(s))(\mathrm{intloc}(0)) = 1$,

(iii) for every read-write integer location a holds $(\mathrm{Initialize}(s))(a) = s(a)$,

(iv) for every f holds $(\mathrm{Initialize}(s))(f) = s(f)$, and

(v) for every l holds $(\mathrm{Initialize}(s))(l) = s(l)$.

(4) s_1 and s_2 are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$ iff $s_1 \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations} \cup \{\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}\}) = s_2 \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations} \cup \{\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}\})$.

(5) If $s_1 \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = s_2 \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$, then $\mathrm{Exec}(i, s_1) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations}) = \mathrm{Exec}(i, s_2) \restriction (\mathrm{Int\text{-}Locations} \cup \mathrm{FinSeq\text{-}Locations})$.

(6) For every parahalting instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{Exec}(i, \mathrm{Initialize}(s)) = \mathrm{IExec}(\mathrm{Macro}(i), s)$.

(7) Let I be a keeping 0 parahalting macro instruction and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(I; j, s))(a) = (\mathrm{Exec}(j, \mathrm{IExec}(I, s)))(a)$.

(8) Let I be a keeping 0 parahalting macro instruction and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(I; j, s))(f) = (\mathrm{Exec}(j, \mathrm{IExec}(I, s)))(f)$.

(9) Let i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(i; j, s))(a) = (\mathrm{Exec}(j, \mathrm{Exec}(i, \mathrm{Initialize}(s))))(a)$.

(10) Let i be a keeping 0 parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and j be a parahalting instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $(\mathrm{IExec}(i; j, s))(f) = (\mathrm{Exec}(j, \mathrm{Exec}(i, \mathrm{Initialize}(s))))(f)$.

4. AN EXAMPLE: SWAP

Let a, b be integer locations. The functor $\mathrm{swap}(a, b)$ yields a macro instruction and is defined as follows:

(Def. 4) $\mathrm{swap}(a, b) = (\mathrm{FirstNotUsed}(\mathrm{Macro}(a{:=}b)){:=}a); (a{:=}b); (b{:=}\mathrm{FirstNotUsed}(\mathrm{Macro}(a{:=}b)))$.

Let a, b be integer locations. Observe that $\mathrm{swap}(a, b)$ is parahalting.

Let a, b be read-write integer locations. Observe that $\mathrm{swap}(a, b)$ is keeping 0.

We now state two propositions:

(11) For all read-write integer locations a, b holds $(\mathrm{IExec}(\mathrm{swap}(a, b), s))(a) = s(b)$ and $(\mathrm{IExec}(\mathrm{swap}(a, b), s))(b) = s(a)$.

(12) $\mathrm{UsedInt}^* \mathrm{Loc}(\mathrm{swap}(a, b)) = \emptyset$.

REFERENCES

[1] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6b.html`.

[2] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6c.html`.

[3] Grzegorz Bancerek. Cardinal numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/card_1.html`.

[4] 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`.

[5] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/funct_7.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. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/cqc_lang.html`.

[8] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/finseq_2.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] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Journal of Formalized Mathematics*, 4, 1992. `http://mizar.org/JFM/Vol4/ami_1.html`.

[11] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\rm FSA}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/sf_mastr.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] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\rm FSA}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_4.html`.

[15] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6a.html`.

[16] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\rm FSA}$ computer. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_2.html`.

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

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

Received July 22, 1996

Published January 2, 2004