FORMALIZED MATHEMATICS
Volume 6, Number 1, 1997
Warsaw University - 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 [27] 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`.

The papers [23], [31], [15], [4], [29], [18], [32], [10], [11], [5], [24], [9], [30], [13], [3], [21], [8], [14], [12], [22], [16], [17], [26], [6], [20], [7], [28], [25], [27], [19], and [1] provide the notation and terminology for this paper.

1. Preliminaries

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

The following propositions are true:

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

ISSN 1426–2630

(1) Let I be a keeping 0 parahalting macro instruction and let 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 let 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) Macro(i) is parahalting.

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

(Def. 2) Macro(i) is keeping 0.

Let us observe that $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ is keeping 0 and parahalting.

Let us note 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 Macro($i$) is parahalting.

Let i be a keeping 0 instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that 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. Note 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. Note 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 verify 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. Note 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. Observe that $b{:=}f_a$ is keeping 0.

Let f be a finite sequence location and let b be a read-write integer location. Observe that b:=len$f$ is keeping 0.

Let i be a parahalting instruction of $\mathbf{SCM}_{\rm FSA}$ and let J be a parahalting macro instruction. One can verify that i;J is parahalting.

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

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

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

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

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

3. Semantics of compositions

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

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

The following propositions are true:

(3) (i) $\quad$ $\mathbf{IC}_{\mathrm{Initialize}(s)} = \mathrm{insloc}(0)$,
(ii) $\quad$ $(\mathrm{Initialize}(s))(\mathrm{intloc}(0)) = 1$,
(iii) $\quad$ for every read-write integer location a holds $(\mathrm{Initialize}(s))(a) = s(a)$,
(iv) $\quad$ for every f holds $(\mathrm{Initialize}(s))(f) = s(f)$, and
(v) $\quad$ for every l holds $(\mathrm{Initialize}(s))(l) = s(l)$.

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

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

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

(8) $\quad$ Let I be a keeping 0 parahalting macro instruction and let j be a parahalting instruction of $\mathbf{SCM}_{\rm FSA}$. 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 let 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 let 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. Note 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. *Formalized Mathematics*, 6(**1**):41–47, 1997.

[2] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part III. *Formalized Mathematics*, 6(**1**):53–57, 1997.

[3] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(**2**):377–382, 1990.

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

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

[6] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **SCM**. *Formalized Mathematics*, 4(**1**):61–67, 1993.

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

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

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

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

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

[12] 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.

[13] Czesław Byliński. Partial functions. *Formalized Mathematics*, 1(**2**):357–367, 1990.

[14] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.

[15] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[16] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[17] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[18] Jan Popiołek. Some properties of functions modul and signum. *Formalized Mathematics*, 1(**2**):263–264, 1990.
[19] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\rm FSA}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[20] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[21] Andrzej Trybulec. Binary operations applied to functions. *Formalized Mathematics*, 1(**2**):329–334, 1990.
[22] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[23] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[24] Andrzej Trybulec and Agata Darmochwał. Boolean domains. *Formalized Mathematics*, 1(**1**):187–190, 1990.
[25] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\rm FSA}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[26] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[27] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
[28] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\rm FSA}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[29] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[30] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[31] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[32] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received July 22, 1996