JOURNAL OF FORMALIZED MATHEMATICS
Volume 10, Released 1998, Published 2003
Inst. of Computer Science, Univ. of Białystok

Another `times` Macro Instruction

Piotr Rudnicki
University of Alberta
Edmonton

Summary. The semantics of the `times` macro is given in [2] only for the case when the body of the macro is parahalting. We remedy this by defining a new `times` macro instruction in terms of `while` (see [11], [14]). The semantics of the new `times` macro is given in a way analogous to the semantics of `while` macros. The new `times` uses an anonymous variable to control the number of its executions. We present two examples: a trivial one and a remake of the macro for the Fibonacci sequence (see [13]).

MML Identifier: `SFMASTR2`.

WWW: `http://mizar.org/JFM/Vol10/sfmastr2.html`

The articles [16], [21], [17], [6], [5], [22], [8], [9], [10], [7], [12], [18], [20], [19], [3], [15], [4], [1], [11], and [13] provide the notation and terminology for this paper.

1. $\mathbf{SCM}_{\mathrm{FSA}}$ Preliminaries

For simplicity, we follow the rules: s, s_1, s_2 denote states of $\mathbf{SCM}_{\mathrm{FSA}}$, a, b denote integer locations, d denotes a read-write integer location, f denotes a finite sequence location, I denotes a macro instruction, J denotes a good macro instruction, and k denotes a natural number.

One can prove the following propositions:

(1) If I is closed on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$ and $b \notin \mathrm{UsedIntLoc}(I)$, then $(\mathrm{IExec}(I,s))(b) = (\mathrm{Initialize}(s))(b)$.

(2) If I is closed on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$ and $f \notin \mathrm{UsedInt}^*\mathrm{Loc}(I)$, then $(\mathrm{IExec}(I,s))(f) = (\mathrm{Initialize}(s))(f)$.

(3) Suppose I is closed on $\mathrm{Initialize}(s)$, halting on $\mathrm{Initialize}(s)$, and parahalting but $s(\mathrm{intloc}(0)) = 1$ or a is read-write but $a \notin \mathrm{UsedIntLoc}(I)$. Then $(\mathrm{IExec}(I,s))(a) = s(a)$.

(4) If $s(\mathrm{intloc}(0)) = 1$, then I is closed on s iff I is closed on $\mathrm{Initialize}(s)$.

(5) If $s(\mathrm{intloc}(0)) = 1$, then I is closed on s and halting on s iff I is closed on $\mathrm{Initialize}(s)$ and halting on $\mathrm{Initialize}(s)$.

(6) Let I_1 be a subset of Int-Locations and F_1 be a subset of FinSeq-Locations. Then $s_1 \upharpoonright (I_1 \cup F_1) = s_2 \upharpoonright (I_1 \cup F_1)$ if and only if the following conditions are satisfied:

(i) for every integer location x such that $x \in I_1$ holds $s_1(x) = s_2(x)$, and

(ii) for every finite sequence location x such that $x \in F_1$ holds $s_1(x) = s_2(x)$.

(7) Let I_1 be a subset of Int-Locations. Then $s_1 \upharpoonright (I_1 \cup \text{FinSeq-Locations}) = s_2 \upharpoonright (I_1 \cup \text{FinSeq-Locations})$ if and only if the following conditions are satisfied:

(i) for every integer location x such that $x \in I_1$ holds $s_1(x) = s_2(x)$, and

(ii) for every finite sequence location x holds $s_1(x) = s_2(x)$.

2. Another times Macro Instruction

Let a be an integer location and let I be a macro instruction. The functor $\text{times}(a,I)$ yielding a macro instruction is defined as follows:

(Def. 1) $\text{times}(a,I) = (a_1{:=}a)$; (**while** $a_1 > 0$ **do** (I; $\text{SubFrom}(a_1, \text{intloc}(0))$)), where $a_1 = 1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(I))$.

We introduce $a \,\text{times}\, I$ as a synonym of $\text{times}(a,I)$.

We now state two propositions:

(8) $\{b\} \cup \text{UsedIntLoc}(I) \subseteq \text{UsedIntLoc}(\text{times}(b,I))$.

(9) $\text{UsedInt}^* \text{Loc}(\text{times}(b,I)) = \text{UsedInt}^* \text{Loc}(I)$.

Let I be a good macro instruction and let a be an integer location. Observe that $\text{times}(a,I)$ is good.

Let s be a state of $\mathbf{SCM}_{\text{FSA}}$, let I be a macro instruction, and let a be an integer location. The functor $\text{StepTimes}(a,I,s)$ yielding a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\text{FSA}}$) is defined as follows:

(Def. 2) $\text{StepTimes}(a,I,s) = \mathit{StepWhile{>}0}(a_1, I; \text{SubFrom}(a_1, \text{intloc}(0)), \text{Exec}(a_1{:=}a, \text{Initialize}(s)))$, where $a_1 = 1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(I))$.

Next we state several propositions:

(10) $(\text{StepTimes}(a,J,s))(0)(\text{intloc}(0)) = 1$.

(11) If $s(\text{intloc}(0)) = 1$ or a is read-write, then $(\text{StepTimes}(a,J,s))(0)(1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(J))) = s(a)$.

(12) Suppose $(\text{StepTimes}(a,J,s))(k)(\text{intloc}(0)) = 1$ and J is closed on $(\text{StepTimes}(a,J,s))(k)$ and halting on $(\text{StepTimes}(a,J,s))(k)$. Then $(\text{StepTimes}(a,J,s))(k+1)(\text{intloc}(0)) = 1$ and if $(\text{StepTimes}(a,J,s))(k)(1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(J))) > 0$, then $(\text{StepTimes}(a,J,s))(k+1)(1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(J))) = (\text{StepTimes}(a,J,s))(k)(1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(J))) - 1$.

(13) If $s(\text{intloc}(0)) = 1$ or a is read-write, then $(\text{StepTimes}(a,I,s))(0)(a) = s(a)$.

(14) $(\text{StepTimes}(a,I,s))(0)(f) = s(f)$.

Let s be a state of $\mathbf{SCM}_{\text{FSA}}$, let a be an integer location, and let I be a macro instruction. We say that ProperTimesBody a, I, s if and only if:

(Def. 3) For every natural number k such that $k < s(a)$ holds I is closed on $(\text{StepTimes}(a,I,s))(k)$ and halting on $(\text{StepTimes}(a,I,s))(k)$.

One can prove the following propositions:

(15) If I is parahalting, then ProperTimesBody a, I, s.

(16) If ProperTimesBody a, J, s, then for every k such that $k \leq s(a)$ holds $(\text{StepTimes}(a,J,s))(k)(\text{intloc}(0)) = 1$.

(17) Suppose $s(\text{intloc}(0)) = 1$ or a is read-write but ProperTimesBody a, J, s. Let given k. If $k \leq s(a)$, then $(\text{StepTimes}(a,J,s))(k)(1^{\text{st}}\text{-RWNotIn}(\{a\} \cup \text{UsedIntLoc}(J))) + k = s(a)$.

(18) Suppose ProperTimesBody a, J, s but $0 \leq s(a)$ but $s(\mathrm{intloc}(0)) = 1$ or a is read-write. Let given k. If $k \geq s(a)$, then $(\mathrm{StepTimes}(a,J,s))(k)(1^{\mathrm{st}}\text{-RWNotIn}(\{a\} \cup \mathrm{UsedIntLoc}(J))) = 0$ and $(\mathrm{StepTimes}(a,J,s))(k)(\mathrm{intloc}(0)) = 1$.

(19) If $s(\mathrm{intloc}(0)) = 1$, then $(\mathrm{StepTimes}(a,I,s))(0)\restriction(\mathrm{UsedIntLoc}(I) \cup \text{FinSeq-Locations}) = s\restriction(\mathrm{UsedIntLoc}(I) \cup \text{FinSeq-Locations})$.

(20) Suppose $(\mathrm{StepTimes}(a,J,s))(k)(\mathrm{intloc}(0)) = 1$ and J is halting on $\mathrm{Initialize}((\mathrm{StepTimes}(a,J,s))(k))$ and closed on $\mathrm{Initialize}((\mathrm{StepTimes}(a,J,s))(k))$ and $(\mathrm{StepTimes}(a,J,s))(k)(1^{\mathrm{st}}\text{-RWNotIn}(\{a\} \cup \mathrm{UsedIntLoc}(J))) > 0$. Then $(\mathrm{StepTimes}(a,J,s))(k+1)\restriction(\mathrm{UsedIntLoc}(J) \cup \text{FinSeq-Locations}) = \mathrm{IExec}(J, (\mathrm{StepTimes}(a,J,s))(k))\restriction(\mathrm{UsedIntLoc}(J) \cup \text{FinSeq-Locations})$.

(21) Suppose ProperTimesBody a, J, s or J is parahalting but $k < s(a)$ but $s(\mathrm{intloc}(0)) = 1$ or a is read-write. Then $(\mathrm{StepTimes}(a,J,s))(k+1)\restriction(\mathrm{UsedIntLoc}(J) \cup \text{FinSeq-Locations}) = \mathrm{IExec}(J, (\mathrm{StepTimes}(a,J,s))(k))\restriction(\mathrm{UsedIntLoc}(J) \cup \text{FinSeq-Locations})$.

(22) If $s(a) \leq 0$ and $s(\mathrm{intloc}(0)) = 1$, then $\mathrm{IExec}(\mathrm{times}(a,I),s)\restriction(\mathrm{UsedIntLoc}(I) \cup \text{FinSeq-Locations}) = s\restriction(\mathrm{UsedIntLoc}(I) \cup \text{FinSeq-Locations})$.

(23) Suppose $s(a) = k$ but ProperTimesBody a, J, s or J is parahalting but $s(\mathrm{intloc}(0)) = 1$ or a is read-write. Then $\mathrm{IExec}(\mathrm{times}(a,J),s)\restriction D = (\mathrm{StepTimes}(a,J,s))(k)\restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(24) If $s(\mathrm{intloc}(0)) = 1$ and if ProperTimesBody a, J, s or J is parahalting, then $\mathrm{times}(a,J)$ is closed on s and $\mathrm{times}(a,J)$ is halting on s.

3. A Trivial Example

Let d be a read-write integer location. The functor triv-times(d) yields a macro instruction and is defined by:

(Def. 4) $\text{triv-times}(d) = \mathrm{times}(d, (\textbf{while } d = 0 \textbf{ do } \mathrm{Macro}(d{:=}d)); \mathrm{SubFrom}(d, \mathrm{intloc}(0)))$.

We now state two propositions:

(25) If $s(d) \leq 0$, then $(\mathrm{IExec}(\text{triv-times}(d),s))(d) = s(d)$.

(26) If $0 \leq s(d)$, then $(\mathrm{IExec}(\text{triv-times}(d),s))(d) = 0$.

4. A Macro for the Fibonacci Sequence

Let N, r_1 be integer locations. The functor Fib-macro(N, r_1) yielding a macro instruction is defined as follows:

(Def. 5) $\text{Fib-macro}(N,r_1) = (N_1{:=}N); \mathrm{SubFrom}(r_1,r_1); (n_1{:=}\mathrm{intloc}(0)); \mathrm{times}(N, \mathrm{AddTo}(r_1,n_1); \mathrm{swap}(r_1,n_1)); (N{:=}N_1)$, where $N_1 = 1^{\mathrm{st}}\text{-NotUsed}(\mathrm{times}(N, \mathrm{AddTo}(r_1,n_1); \mathrm{swap}(r_1,n_1)))$ and $n_1 = 1^{\mathrm{st}}\text{-RWNotIn}(\{N, r_1\})$.

Next we state the proposition

(27) Let N, r_1 be read-write integer locations. Suppose $N \neq r_1$. Let n be a natural number. If $n = s(N)$, then $(\mathrm{IExec}(\text{Fib-macro}(N,r_1),s))(r_1) = \mathrm{Fib}(n)$ and $(\mathrm{IExec}(\text{Fib-macro}(N,r_1),s))(N) = s(N)$.

References

[1] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa7b.html.

[2] Noriko Asamoto. The loop and times macroinstruction for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 9, 1997. http://mizar.org/JFM/Vol9/scmfsa8c.html.

[3] 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.

[4] 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`.

[5] Grzegorz Bancerek. The fundamental properties of natural numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/nat_1.html`.

[6] Grzegorz Bancerek. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/card_3.html`.

[7] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part I - preliminaries. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/pre_ff.html`.

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

[9] Czesław Byliński. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funct_2.html`.

[10] 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`.

[11] Jing-Chao Chen. While macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 9, 1997. `http://mizar.org/JFM/Vol9/scmfsa_9.html`.

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

[13] Piotr Rudnicki. On the composition of non-parahalting macro instructions. *Journal of Formalized Mathematics*, 10, 1998. `http://mizar.org/JFM/Vol10/sfmastr1.html`.

[14] Piotr Rudnicki. The `while` macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Journal of Formalized Mathematics*, 10, 1998. `http://mizar.org/JFM/Vol10/scmfsa9a.html`.

[15] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/sf_mastr.html`.

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

[17] Andrzej Trybulec. Subsets of real numbers. *Journal of Formalized Mathematics*, Addenda, 2003. `http://mizar.org/JFM/Addenda/numbers.html`.

[18] 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`.

[19] 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`.

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

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

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

Received June 4, 1998

Published January 2, 2004