JOURNAL OF FORMALIZED MATHEMATICS
Volume 11, Released 1999, Published 2003
Inst. of Computer Science, Univ. of Białystok

Insert Sort on $\mathbf{SCM}_{\mathrm{FSA}}$[1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. This article describes the insert sorting algorithm using macro instructions such as if-Macro (conditional branch macro instructions), for-loop macro instructions and While-Macro instructions etc. From the viewpoint of initialization, we generalize the halting and computing problem of the While-Macro. Generally speaking, it is difficult to judge whether the While-Macro is halting or not by way of loop inspection. For this reason, we introduce a practical and simple method, called body-inspection. That is, in many cases, we can prove the halting problem of the While-Macro by only verifying the nature of the body of the While-Macro, rather than the While-Macro itself. In fact, we have used this method in justifying the halting of the insert sorting algorithm. Finally, we prove that the insert sorting algorithm given in the article is autonomic and its computing result is correct.

MML Identifier: `SCMISORT`.

WWW: `http://mizar.org/JFM/Vol11/scmisort.html`

The articles [25], [24], [35], [26], [8], [15], [36], [13], [14], [16], [12], [7], [10], [9], [21], [28], [11], [23], [31], [29], [30], [22], [5], [6], [3], [1], [17], [2], [32], [34], [18], [27], [4], [20], [19], and [33] provide the notation and terminology for this paper.

1. Preliminaries

Let i be a good instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that $\mathrm{Macro}(i)$ is good.

Let a be a read-write integer location and let b be an integer location. One can check that $\mathrm{AddTo}(a,b)$ is good.

One can prove the following four propositions:

(1) For every function f and for all sets d, r such that $d \in \mathrm{dom}\, f$ holds $\mathrm{dom}\, f = \mathrm{dom}(f+\cdot(d \dot{\longmapsto} r))$.

(2) Let p be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$, l be an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$, and i_1 be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $l \in \mathrm{dom}\, p$ and there exists an instruction p_1 of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $p_1 = p(l)$ and $\mathrm{UsedIntLoc}(p_1) = \mathrm{UsedIntLoc}(i_1)$. Then $\mathrm{UsedIntLoc}(p) = \mathrm{UsedIntLoc}(p+\cdot(l \dot{\longmapsto} i_1))$.

(3) For every integer location a and for every macro instruction I holds $(\mathbf{if}\, a > 0\ \mathbf{then}\ I;\ \mathrm{Goto}(\mathrm{insloc}(0))\ \mathbf{else}\ (\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}))(\mathrm{insloc}(\mathrm{card}\, I+4)) = \mathrm{goto}\ \mathrm{insloc}(\mathrm{card}\, I+4)$.

(4) Let p be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$, l be an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$, and i_1 be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose $l \in \mathrm{dom}\, p$ and there exists an instruction p_1 of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $p_1 = p(l)$ and $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(p_1) = \mathrm{UsedInt}^{*}\,\mathrm{Loc}(i_1)$. Then $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(p) = \mathrm{UsedInt}^{*}\,\mathrm{Loc}(p+\cdot(l \dot{\longmapsto} i_1))$.

[1]This research is supported by the National Natural Science Foundation of China Grant No. 69873033.

For simplicity, we adopt the following convention: s is a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I is a macro instruction, a is a read-write integer location, and j, k, n are natural numbers.

We now state a number of propositions:

(6)[1] For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every macro instruction I such that $s(\mathrm{intloc}(0)) = 1$ and $\mathbf{IC}_s = \mathrm{insloc}(0)$ holds $s{+}{\cdot}I = s{+}{\cdot}\,\mathrm{Initialized}(I)$.

(7) Let I be a macro instruction and a, b be integer locations. If I does not destroy b, then **while** $a > 0$ **do** I does not destroy b.

(8) If $n \leq 11$, then $n = 0$ or $n = 1$ or $n = 2$ or $n = 3$ or $n = 4$ or $n = 5$ or $n = 6$ or $n = 7$ or $n = 8$ or $n = 9$ or $n = 10$ or $n = 11$.

(9) Let f, g be finite sequences of elements of $\mathbb{Z}$ and m, n be natural numbers. Suppose $1 \leq n$ and $n \leq \mathrm{len}\, f$ and $1 \leq m$ and $m \leq \mathrm{len}\, f$ and $g = f{+}{\cdot}(m, f_n){+}{\cdot}(n, f_m)$. Then

(i) $f(m) = g(n)$,

(ii) $f(n) = g(m)$,

(iii) for every set k such that $k \neq m$ and $k \neq n$ and $k \in \mathrm{dom}\, f$ holds $f(k) = g(k)$, and

(iv) f and g are fiberwise equipotent.

(10) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$ and I be a macro instruction. Suppose I is halting on Initialize(s). Let a be an integer location. Then $(\mathrm{IExec}(I,s))(a) = (\mathrm{Computation}(\mathrm{Initialize}(s){+}{\cdot}(I{+}{\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(\mathrm{LifeSpan}(\mathrm{Initialize}(s){+}{\cdot}(I{+}{\cdot}\,\mathrm{Start\text{-}At}(\mathrm{insloc}(0)))))(a)$.

(11) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$ and I be an InitHalting macro instruction. Suppose $\mathrm{Initialized}(I) \subseteq s_1$ and $\mathrm{Initialized}(I) \subseteq s_2$ and s_1 and s_2 are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Let k be a natural number. Then $(\mathrm{Computation}(s_1))(k)$ and $(\mathrm{Computation}(s_2))(k)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$ and $\mathrm{CurInstr}((\mathrm{Computation}(s_1))(k)) = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(k))$.

(12) Let s_1, s_2 be states of $\mathbf{SCM}_{\mathrm{FSA}}$ and I be an InitHalting macro instruction. Suppose $\mathrm{Initialized}(I) \subseteq s_1$ and $\mathrm{Initialized}(I) \subseteq s_2$ and s_1 and s_2 are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $\mathrm{LifeSpan}(s_1) = \mathrm{LifeSpan}(s_2)$ and $\mathrm{Result}(s_1)$ and $\mathrm{Result}(s_2)$ are equal outside the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(13) For every macro instruction I and for every finite sequence location f holds $f \notin \mathrm{dom} I$.

(14) For every macro instruction I and for every integer location a holds $a \notin \mathrm{dom} I$.

(15) Let N be a non empty set with non empty elements, S be a halting IC-Ins-separated definite non empty non void AMI over N, and s be a state of S. If $\mathrm{LifeSpan}(s) \leq j$ and s is halting, then $(\mathrm{Computation}(s))(j) = (\mathrm{Computation}(s))(\mathrm{LifeSpan}(s))$.

2. Basic Property of **while** Macro

One can prove the following propositions:

(16) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. Suppose $s(a) \leq 0$. Then **while** $a > 0$ **do** I is halting onInit s and **while** $a > 0$ **do** I is closed onInit s.

(17) Let a be an integer location, I be a macro instruction, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k be a natural number. Suppose that

(i) I is closed onInit s,

(ii) I is halting onInit s,

(iii) $k < \mathrm{LifeSpan}(s{+}{\cdot}\,\mathrm{Initialized}(I))$,

[1] The proposition (5) has been removed.

(iv) $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+k)} = \mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(I)))(k)} + 4$, and

(v) $(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+k)\restriction D = (\text{Computation}(s+\cdot \text{Initialized}(I)))(k)\restriction D$.

Then $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+k+1)} = \mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(I)))(k+1)} + 4$ and $(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+k+1)\restriction D = (\text{Computation}(s+\cdot \text{Initialized}(I)))(k+1)\restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(18) Let a be an integer location, I be a macro instruction, and s be a state of $\mathbf{SCM}_{\rm FSA}$. Suppose I is closed onInit s and I is halting onInit s and $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+\text{LifeSpan}(s+\cdot \text{Initialized}(I)))} = \mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I)))} +$ 4. Then $\text{CurInstr}((\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(1+\text{LifeSpan}(s+\cdot \text{Initialized}(I)))) =$ goto $\text{insloc}(\text{card}\, I+4)$.

(19) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a macro instruction, and a be a read-write integer location. Suppose I is closed onInit s and I is halting onInit s and $s(a) > 0$. Then $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I))+3)} = \text{insloc}(0)$ and for every natural number k such that $k \leq \text{LifeSpan}(s+\cdot \text{Initialized}(I)) + 3$ holds $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(k)} \in \text{dom}(\textbf{while } a>0 \textbf{ do } I)$.

(20) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a macro instruction, and a be a read-write integer location. Suppose I is closed onInit s and I is halting onInit s and $s(a) > 0$. Let k be a natural number. If $k \leq \text{LifeSpan}(s+\cdot \text{Initialized}(I)) + 3$, then $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(k)} \in \text{dom}(\textbf{while } a>0 \textbf{ do } I)$.

(21) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a macro instruction, and a be a read-write integer location. Suppose I is closed onInit s and I is halting onInit s and $s(a) > 0$. Then $\mathbf{IC}_{(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I))+3)} = \text{insloc}(0)$ and $(\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I))+3)\restriction D = (\text{Computation}(s+\cdot \text{Initialized}(I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I)))\restriction D$, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(22) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be an InitHalting macro instruction, and a be a read-write integer location. Suppose $s(a) > 0$. Then there exists a state s_2 of $\mathbf{SCM}_{\rm FSA}$ and there exists a natural number k such that

(i) $s_2 = s+\cdot \text{Initialized}(\textbf{while } a>0 \textbf{ do } I)$,

(ii) $k = \text{LifeSpan}(s+\cdot \text{Initialized}(I)) + 3$,

(iii) $\mathbf{IC}_{(\text{Computation}(s_2))(k)} = \text{insloc}(0)$,

(iv) for every integer location b holds $(\text{Computation}(s_2))(k)(b) = (\text{IExec}(I,s))(b)$, and

(v) for every finite sequence location f holds $(\text{Computation}(s_2))(k)(f) = (\text{IExec}(I,s))(f)$.

Let us consider s, I, a. The functor $\textit{StepWhile>0}(a,s,I)$ yielding a function from $\mathbb{N}$ into $\prod$(the object kind of $\mathbf{SCM}_{\rm FSA}$) is defined by the conditions (Def. 1).

(Def. 1)(i) $(\textit{StepWhile>0}(a,s,I))(0) = s$ **qua** element of $\prod$(the object kind of $\mathbf{SCM}_{\rm FSA}$) **qua** non empty set, and

(ii) for every natural number i holds $(\textit{StepWhile>0}(a,s,I))(i+1) = (\text{Computation}((\textit{StepWhile>0}(a,s,I))(i)+\cdot \text{Initialize}$ 0 **do** $I)))(\text{LifeSpan}((\textit{StepWhile>0}(a,s,I))(i)+\cdot \text{Initialized}(I)) + 3)$.

Next we state several propositions:

(25)[2] $(\textit{StepWhile>0}(a,s,I))(k+1) = (\textit{StepWhile>0}(a,(\textit{StepWhile>0}(a,s,I))(k),I))(1)$.

(26) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\rm FSA}$. Then $(\textit{StepWhile>0}(a,s,I))(0+1) = (\text{Computation}(s+\cdot \text{Initialized}(\textbf{while } a >$ 0 **do** $I)))(\text{LifeSpan}(s+\cdot \text{Initialized}(I)) + 3)$.

[2] The propositions (23) and (24) have been removed.

(27) Let I be a macro instruction, a be a read-write integer location, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and k, n be natural numbers. Suppose $\mathbf{IC}_{(\mathit{StepWhile{>}0}(a,s,I))(k)} = \mathrm{insloc}(0)$ and $(\mathit{StepWhile{>}0}(a,s,I))(k) = (\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(\mathbf{while}\ a > 0\ \mathbf{do}\ I)))(n)$ and $(\mathit{StepWhile{>}0}(a,s,I))(k)(\mathrm{intloc}(0)) = 1$. Then $(\mathit{StepWhile{>}0}(a,s,I))(k) = (\mathit{StepWhile{>}0}(a,s,I))(k){+\cdot}\,\mathrm{Initialized}($**wh**
0 **do** $I)$ and $(\mathit{StepWhile{>}0}(a,s,I))(k+1) = (\mathrm{Computation}(s{+\cdot}\,\mathrm{Initialized}(\mathbf{while}\ a > 0\ \mathbf{do}\ I)))(n+ (\mathrm{LifeSpan}((\mathit{StepWhile{>}0}(a,s,I))(k){+\cdot}\,\mathrm{Initialized}(I)) + 3))$.

(28) Let I be a macro instruction, a be a read-write integer location, and s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{N}$ such that let k be a natural number. Then

(i) if $f((\mathit{StepWhile{>}0}(a,s,I))(k)) \neq 0$, then $f((\mathit{StepWhile{>}0}(a,s,I))(k+1)) < f((\mathit{StepWhile{>}0}(a,s,I))(k))$ and I is closed onInit $(\mathit{StepWhile{>}0}(a,s,I))(k)$ and I is halting onInit $(\mathit{StepWhile{>}0}(a,s,I))(k)$,

(ii) $(\mathit{StepWhile{>}0}(a,s,I))(k+1)(\mathrm{intloc}(0)) = 1$, and

(iii) $f((\mathit{StepWhile{>}0}(a,s,I))(k)) = 0$ iff $(\mathit{StepWhile{>}0}(a,s,I))(k)(a) \leq 0$.

Then **while** $a > 0$ **do** I is halting onInit s and **while** $a > 0$ **do** I is closed onInit s.

(29) Let I be a good InitHalting macro instruction and a be a read-write integer location. Suppose that for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $s(a) > 0$ holds $(\mathrm{IExec}(I,s))(a) < s(a)$. Then **while** $a > 0$ **do** I is InitHalting.

(30) Let I be a good InitHalting macro instruction and a be a read-write integer location. Suppose that for every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $(\mathrm{IExec}(I,s))(a) < s(a)$ or $(\mathrm{IExec}(I,s))(a) \leq 0$. Then **while** $a > 0$ **do** I is InitHalting.

Let D be a set, let f be a function from D into $\mathbb{Z}$, and let d be an element of D. Then $f(d)$ is an integer.

We now state several propositions:

(31) Let I be a good InitHalting macro instruction and a be a read-write integer location. Given a function f from $\prod$(the object kind of $\mathbf{SCM}_{\mathrm{FSA}}$) into $\mathbb{Z}$ such that let s, t be states of $\mathbf{SCM}_{\mathrm{FSA}}$. Then

(i) if $f(s) > 0$, then $f(\mathrm{IExec}(I,s)) < f(s)$,

(ii) if $s{\upharpoonright}D = t{\upharpoonright}D$, then $f(s) = f(t)$, and

(iii) $f(s) \leq 0$ iff $s(a) \leq 0$.

Then **while** $a > 0$ **do** I is InitHalting, where $D = \text{Int-Locations} \cup \text{FinSeq-Locations}$.

(32) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, and a be a read-write integer location. If $s(a) \leq 0$, then $\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s){\upharpoonright}(\text{Int-Locations} \cup \text{FinSeq-Locations}) = \mathrm{Initialize}(s){\upharpoonright}(\text{Int-Locations} \cup \text{FinSeq-Locations})$.

(33) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a good InitHalting macro instruction, and a be a read-write integer location. If $s(a) > 0$ and **while** $a > 0$ **do** I is InitHalting, then $\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s){\upharpoonright}(\text{Int-Locations} \cup \text{FinSeq-Locations}) = \mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, \mathrm{IExec}(I,s)){\upharpoonright}(\text{Int-Locations} \cup \text{FinSeq-Locations})$.

(34) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, f be a finite sequence location, and a be a read-write integer location. If $s(a) \leq 0$, then $(\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s))(f) = s(f)$.

(35) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a macro instruction, b be an integer location, and a be a read-write integer location. If $s(a) \leq 0$, then $(\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s))(b) = (\mathrm{Initialize}(s))(b)$.

(36) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a good InitHalting macro instruction, f be a finite sequence location, and a be a read-write integer location. If $s(a) > 0$ and **while** $a > 0$ **do** I is InitHalting, then $(\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s))(f) = (\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, \mathrm{IExec}(I,s)))(f)$.

(37) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, I be a good InitHalting macro instruction, b be an integer location, and a be a read-write integer location. If $s(a) > 0$ and **while** $a > 0$ **do** I is InitHalting, then $(\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, s))(b) = (\mathrm{IExec}(\mathbf{while}\ a > 0\ \mathbf{do}\ I, \mathrm{IExec}(I,s)))(b)$.

3. Insert Sort Algorithm

Let f be a finite sequence location. The functor insert-sort f yielding a macro instruction is defined by:

(Def. 2) insert-sort $f = i_2$; $(a_1{:=}\mathrm{len}\, f)$; $\mathrm{SubFrom}(a_1, a_0)$; $\mathrm{Times}(a_1, (a_2{:=}\mathrm{len}\, f)$; $\mathrm{SubFrom}(a_2, a_1)$; $(a_3{:=}a_2)$; $\mathrm{AddTo}(a_3, a_0)$; 0 **do** $((a_5{:=}f_{a_2})$; $\mathrm{SubFrom}(a_5, a_6)$; (**if** $a_5 > 0$ **then** $\mathrm{Macro}(\mathrm{SubFrom}(a_2, a_2))$ **else** $(\mathrm{AddTo}(a_4, a_0)$; $\mathrm{SubFrom}(a_2, a_0)))))$ where $i_2 = (a_2{:=}a_0)$; $(a_3{:=}a_0)$; $(a_4{:=}a_0)$; $(a_5{:=}a_0)$; $(a_6{:=}a_0)$, $a_2 = \mathrm{intloc}(2)$, $a_0 = \mathrm{intloc}(0)$, $a_3 = \mathrm{intloc}(3)$, $a_4 = \mathrm{intloc}(4)$, $a_5 = \mathrm{intloc}(5)$, $a_6 = \mathrm{intloc}(6)$, and $a_1 = \mathrm{intloc}(1)$.

The macro instruction Insert-Sort-Algorithm is defined by:

(Def. 3) Insert-Sort-Algorithm = insert-sort fsloc(0).

The following propositions are true:

(38) For every finite sequence location f holds UsedIntLoc(insert-sort $f) = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$, where $a_0 = \mathrm{intloc}(0)$, $a_1 = \mathrm{intloc}(1)$, $a_2 = \mathrm{intloc}(2)$, $a_3 = \mathrm{intloc}(3)$, $a_4 = \mathrm{intloc}(4)$, $a_5 = \mathrm{intloc}(5)$, and $a_6 = \mathrm{intloc}(6)$.

(39) For every finite sequence location f holds UsedInt*Loc(insert-sort $f) = \{f\}$.

(40) For all instructions k_1, k_2, k_3, k_4 of $\mathbf{SCM}_{\rm FSA}$ holds $\mathrm{card}(k_1; k_2; k_3; k_4) = 8$.

(41) For all instructions k_1, k_2, k_3, k_4, k_5 of $\mathbf{SCM}_{\rm FSA}$ holds $\mathrm{card}(k_1; k_2; k_3; k_4; k_5) = 10$.

(42) For every finite sequence location f holds card insert-sort $f = 82$.

(43) For every finite sequence location f and for every natural number k such that $k < 82$ holds $\mathrm{insloc}(k) \in \mathrm{dom}$ insert-sort f.

One can prove the following proposition

(44) insert-sort fsloc(0) is keepInt0 1 and InitHalting.

We now state several propositions:

(45) Let s be a state of $\mathbf{SCM}_{\rm FSA}$. Then

(i) $s(f_0)$ and (IExec(insert-sort $f_0, s))(f_0)$ are fiberwise equipotent, and

(ii) for all natural numbers i, j such that $i \geq 1$ and $j \leq \mathrm{len}\, s(f_0)$ and $i < j$ and for all integers x_1, x_2 such that $x_1 = (\mathrm{IExec}(\text{insert-sort } f_0, s))(f_0)(i)$ and $x_2 = (\mathrm{IExec}(\text{insert-sort } f_0, s))(f_0)(j)$ holds $x_1 \geq x_2$,

where $f_0 = \mathrm{fsloc}(0)$.

(46) Let i be a natural number, s be a state of $\mathbf{SCM}_{\rm FSA}$, and w be a finite sequence of elements of $\mathbb{Z}$. If Initialized(Insert-Sort-Algorithm)$+\cdot(\mathrm{fsloc}(0) \dot{\longmapsto} w) \subseteq s$, then $\mathbf{IC}_{(\mathrm{Computation}(s))(i)} \in$ dom Insert-Sort-Algorithm.

(47) Let s be a state of $\mathbf{SCM}_{\rm FSA}$ and t be a finite sequence of elements of $\mathbb{Z}$. Suppose Initialized(Insert-Sort-Algorithm)$+\cdot(\mathrm{fsloc}(0) \dot{\longmapsto} t) \subseteq s$. Then there exists a finite sequence u of elements of $\mathbb{R}$ such that

(i) t and u are fiberwise equipotent,

(ii) u is non-increasing and a finite sequence of elements of $\mathbb{Z}$, and

(iii) $(\mathrm{Result}(s))(\mathrm{fsloc}(0)) = u$.

(48) For every finite sequence w of elements of $\mathbb{Z}$ holds Initialized(Insert-Sort-Algorithm)$+\cdot(\mathrm{fsloc}(0) \dot{\longmapsto} w)$ is autonomic.

(49) Initialized(Insert-Sort-Algorithm) computes Sorting-Function.

REFERENCES

[1] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\rm FSA}$. Part I. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa8a.html`.

[2] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\rm FSA}$. Part II. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa8b.html`.

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

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

[5] 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`.

[6] 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`.

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

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

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

[10] 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`.

[11] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **scm**. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/scm_1.html`.

[12] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/funct_7.html`.

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

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

[15] Czesław Byliński. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/cqc_lang.html`.

[16] 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`.

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

[18] Jing-Chao Chen and Yatsuka Nakamura. Bubble sort on $\mathbf{SCM}_{\rm FSA}$. *Journal of Formalized Mathematics*, 10, 1998. `http://mizar.org/JFM/Vol10/scmbsort.html`.

[19] Jing-Chao Chen and Yatsuka Nakamura. Initialization halting concepts and their basic properties of $\mathbf{SCM}_{\rm FSA}$. *Journal of Formalized Mathematics*, 10, 1998. `http://mizar.org/JFM/Vol10/scm_halt.html`.

[20] Jarosław Kotowicz. Functions and finite sequences of real numbers. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/rfinseq.html`.

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

[22] 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`.

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

[24] Andrzej Trybulec. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/enumset1.html`.

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

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

[27] Andrzej Trybulec and Agata Darmochwał. Boolean domains. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/finsub_1.html`.

[28] 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`.

[29] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of **SCM**$_{\text{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_4.html`.

[30] 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`.

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

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

[33] Wojciech A. Trybulec. Groups. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/group_1.html`.

[34] Wojciech A. Trybulec. Pigeon hole principle. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/finseq_4.html`.

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

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

Received March 13, 1999

Published January 2, 2004