FORMALIZED MATHEMATICS
Volume 7, Number 1, 1998
University of Białystok

Bubble Sort on $\mathbf{SCM}_{\rm FSA}$

Jing-Chao Chen
Shanghai Jiaotong University

Yatsuka Nakamura
Shinshu University
Nagano

Summary. We present the bubble sorting algorithm using macro instructions such as the if-Macro (conditional branch macro instructions) and the Times-Macro (for-loop macro instructions) etc. The correctness proof of the program should include the proof of autonomic, halting and the correctness of the program result. In the three terms, we justify rigorously the correctness of the bubble sorting algorithm. In order to prove it is autonomic, we use the following theorem: if all variables used by the program are initialized, it is autonomic. This justification method probably reveals that autonomic concept is not important.

MML Identifier: SCMBSORT.

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

1. Preliminaries

For simplicity, we adopt the following rules: p is a programmed finite partial state of $\mathbf{SCM}_{\rm FSA}$, i_1 is an instruction of $\mathbf{SCM}_{\rm FSA}$, i, j, k are natural numbers, f_1, f are finite sequence locations, a, b, d_1, d_2 are integer locations, l, l_1 are instructions-locations of $\mathbf{SCM}_{\rm FSA}$, and s_1 is a state of $\mathbf{SCM}_{\rm FSA}$.

We now state a number of propositions:

(1) Let I, J be macro instructions and a, b be integer locations. Suppose I does not destroy b and J does not destroy b. Then **if** $a > 0$ **then** I **else** J does not destroy b.

(2) Let I, J be macro instructions and a, b be integer locations. Suppose I does not destroy b and J does not destroy b. Then **if** $a = 0$ **then** I **else** J does not destroy b.

(3) Let I be a macro instruction and a, b be integer locations. If I does not destroy b and $a \neq b$, then $\mathrm{Times}(a, I)$ does not destroy b.

(4) For every function f and for all sets n, m holds $(f{+\cdot}(n \mapsto m){+\cdot}(m \mapsto n))(m) = n$.

(5) For every function f and for all sets n, m holds $(f{+\cdot}(n \mapsto m){+\cdot}(m \mapsto n))(n) = m$.

(6) For every function f and for all sets n, m, x such that $x \in \mathrm{dom}\, f$ and $x \neq m$ and $x \neq n$ holds $(f{+\cdot}(n \mapsto m){+\cdot}(m \mapsto n))(x) = f(x)$.

(7) Let f, g be functions and m, n be sets. Suppose that

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

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

(iii) $m \in \mathrm{dom}\, f$,

(iv) $n \in \mathrm{dom}\, f$,

(v) $\mathrm{dom}\, f = \mathrm{dom}\, g$, and

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

Then f and g are fiberwise equipotent.

(8) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, f be a finite sequence location, and a, b be integer locations. Then $(\mathrm{Exec}(b{:=}f_a, s))(b) = \pi_{|s(a)|} s(f)$.

(9) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, f be a finite sequence location, and a, b be integer locations. Then $(\mathrm{Exec}(f_a{:=}b, s))(f) = s(f) +\cdot (|s(a)|, s(b))$.

(10) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, f be a finite sequence location, m, n be natural numbers, and a be an integer location. If $m \neq n + 1$, then $(\mathrm{Exec}(\mathrm{intloc}(m){:=}f_a, \mathrm{Initialize}(s)))(\mathrm{intloc}(n + 1)) = s(\mathrm{intloc}(n + 1))$.

(11) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, m, n be natural numbers, and a be an integer location. If $m \neq n{+}1$, then $(\mathrm{Exec}(\mathrm{intloc}(m){:=}a, \mathrm{Initialize}(s)))(\mathrm{intloc}(n{+}1)) = s(\mathrm{intloc}(n + 1))$.

(12) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, f be a finite sequence location, and a be a read-write integer location. Then $(\mathrm{IExec}(\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, s))(a) = s(a)$ and $(\mathrm{IExec}(\mathrm{Stop}_{\mathrm{SCM}_{\mathrm{FSA}}}, s))(f) = s(f)$.

In the sequel n denotes a natural number.

One can prove the following propositions:

(13) If $n \leqslant 10$, 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 $n = 10$.

(14) Suppose $n \leqslant 12$. 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 $n = 11$ or $n = 12$.

(15) Let f, g be functions and X be a set. If $\operatorname{dom} f = \operatorname{dom} g$ and for every set x such that $x \in X$ holds $f(x) = g(x)$, then $f{\restriction}X = g{\restriction}X$.

(16) If $i_1 \in \operatorname{rng} p$ and if $i_1 = a{:=}b$ or $i_1 = \mathrm{AddTo}(a, b)$ or $i_1 = \mathrm{SubFrom}(a, b)$ or $i_1 = \mathrm{MultBy}(a, b)$ or $i_1 = \mathrm{Divide}(a, b)$, then $a \in \mathrm{UsedIntLoc}(p)$ and $b \in \mathrm{UsedIntLoc}(p)$.

(17) If $i_1 \in \operatorname{rng} p$ and if $i_1 = \mathbf{if}\ a = 0\ \mathbf{goto}\ l_1$ or $i_1 = \mathbf{if}\ a > 0\ \mathbf{goto}\ l_1$, then $a \in \mathrm{UsedIntLoc}(p)$.

(18) If $i_1 \in \operatorname{rng} p$ and if $i_1 = b{:=}{f_1}_a$ or $i_1 = {f_1}_a{:=}b$, then $a \in \mathrm{UsedIntLoc}(p)$ and $b \in \mathrm{UsedIntLoc}(p)$.

(19) If $i_1 \in \operatorname{rng} p$ and if $i_1 = b{:=}{f_1}_a$ or $i_1 = {f_1}_a{:=}b$, then $f_1 \in \mathrm{UsedInt}^*\,\mathrm{Loc}(p)$.

(20) If $i_1 \in \operatorname{rng} p$ and if $i_1 = a{:=}\mathrm{len} f_1$ or $i_1 = f_1{:=}\langle\underbrace{0, \ldots, 0}_{a}\rangle$, then $a \in \mathrm{UsedIntLoc}(p)$.

(21) If $i_1 \in \operatorname{rng} p$ and if $i_1 = a{:=}\mathrm{len} f_1$ or $i_1 = f_1{:=}\langle\underbrace{0, \ldots, 0}_{a}\rangle$, then $f_1 \in \mathrm{UsedInt}^*\,\mathrm{Loc}(p)$.

(22) Let p be a macro instruction, s_2, s_3 be states of $\mathbf{SCM}_{\mathrm{FSA}}$, and given i. If $p \subseteq s_2$ and $p \subseteq s_3$, then $(\mathrm{Computation}(s_2))(i){\restriction}\operatorname{dom} p = (\mathrm{Computation}(s_3))(i){\restriction}\operatorname{dom} p$.

(23) Let t be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$, p be a macro instruction, and x be a set. Suppose $\operatorname{dom} t \subseteq \text{Int-Locations} \cup \text{FinSeq-Locations}$ and $x \in \operatorname{dom} t \cup \mathrm{UsedInt}^*\,\mathrm{Loc}(p) \cup \mathrm{UsedIntLoc}(p)$. Then x is an integer location or a finite sequence location.

(24) For every f_1 holds $(\mathrm{Exec}(\mathrm{Divide}(d_1, d_2), s_1))(f_1) = s_1(f_1)$ and $(\mathrm{Exec}(\mathrm{Divide}(d_1, d_2), s_1))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_{(s_1)})$.

(25) Let i, k be natural numbers, t be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$, p be a macro instruction, and s_2, s_3 be states of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that

(i) $k \leqslant i$,

(ii) $p \subseteq s_2$,

(iii) $p \subseteq s_3$,

(iv) $\operatorname{dom} t \subseteq \text{Int-Locations} \cup \text{FinSeq-Locations}$,

(v) for every j holds $\mathbf{IC}_{(\mathrm{Computation}(s_2))(j)} \in \operatorname{dom} p$ and $\mathbf{IC}_{(\mathrm{Computation}(s_3))(j)} \in \operatorname{dom} p$,

(vi) $(\mathrm{Computation}(s_2))(k)(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = (\mathrm{Computation}(s_3))(k)(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}})$, and

(vii) $(\mathrm{Computation}(s_2))(k){\restriction}(\operatorname{dom} t \cup \mathrm{UsedInt}^*\,\mathrm{Loc}(p) \cup \mathrm{UsedIntLoc}(p)) = (\mathrm{Computation}(s_3))(k){\restriction}(\operatorname{dom} t \cup \mathrm{UsedInt}^*\,\mathrm{Loc}(p) \cup \mathrm{UsedIntLoc}(p))$.

Then $(\mathrm{Computation}(s_2))(i)(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = (\mathrm{Computation}(s_3))(i)(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}})$ and $(\mathrm{Computation}(s_2))(i){\restriction}(\operatorname{dom} t \cup \mathrm{UsedInt}^*\,\mathrm{Loc}(p) \cup \mathrm{UsedIntLoc}(p)) = (\mathrm{Computation}(s_3))(i){\restriction}(\operatorname{dom} t \cup \mathrm{UsedInt}^*\,\mathrm{Loc}(p) \cup \mathrm{UsedIntLoc}(p))$.

(26) Let i, k be natural numbers, p be a macro instruction, and s_2, s_3 be states of $\mathbf{SCM}_{\rm FSA}$. Suppose $k \leqslant i$ and $p \subseteq s_2$ and $p \subseteq s_3$ and for every j holds $\mathbf{IC}_{({\rm Computation}(s_2))(j)} \in {\rm dom}\, p$ and $\mathbf{IC}_{({\rm Computation}(s_3))(j)} \in {\rm dom}\, p$ and $({\rm Computation}(s_2))(k)(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}) = ({\rm Computation}(s_3))(k)(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}})$ and $({\rm Computation}(s_2))(k){\restriction}({\rm UsedInt}^*\,{\rm Loc}(p) \cup {\rm UsedIntLoc}(p)) =$ $({\rm Computation}(s_3))(k){\restriction}({\rm UsedInt}^*\,{\rm Loc}(p) \cup {\rm UsedIntLoc}(p))$.
Then $({\rm Computation}(s_2))(i)(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}) = ({\rm Computation}(s_3))(i)(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}})$ and $({\rm Computation}(s_2))(i){\restriction}({\rm UsedInt}^*\,{\rm Loc}(p) \cup {\rm UsedIntLoc}(p)) =$ $({\rm Computation}(s_3))(i){\restriction}({\rm UsedInt}^*\,{\rm Loc}(p) \cup {\rm UsedIntLoc}(p))$.

(27) ${\rm UsedIntLoc}({\rm Stop}_{{\rm SCM}_{\rm FSA}}) = \emptyset$.

(28) ${\rm UsedIntLoc}({\rm Goto}(l)) = \emptyset$.

(29) For all macro instructions I, J and for every integer location a holds ${\rm UsedIntLoc}(\mathbf{if}\ a = 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \{a\} \cup {\rm UsedIntLoc}(I) \cup {\rm UsedIntLoc}(J)$ and ${\rm UsedIntLoc}(\mathbf{if}\ a > 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = \{a\} \cup {\rm UsedIntLoc}(I) \cup {\rm UsedIntLoc}(J)$.

(30) For every macro instruction I and for every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds ${\rm UsedIntLoc}({\rm Directed}(I, l)) = {\rm UsedIntLoc}(I)$.

(31) For every integer location a and for every macro instruction I holds ${\rm UsedIntLoc}({\rm Times}(a, I)) = {\rm UsedIntLoc}(I) \cup \{a, {\rm intloc}(0)\}$.

(32) For all sets x_1, x_2, x_3 holds $\{x_2, x_1\} \cup \{x_3, x_1\} = \{x_1, x_2, x_3\}$.

(33) ${\rm UsedInt}^*\,{\rm Loc}({\rm Stop}_{{\rm SCM}_{\rm FSA}}) = \emptyset$.

(34) ${\rm UsedInt}^*\,{\rm Loc}({\rm Goto}(l)) = \emptyset$.

(35) For all macro instructions I, J and for every integer location a holds ${\rm UsedInt}^*\,{\rm Loc}(\mathbf{if}\ a = 0\ \mathbf{then}\ I\ \mathbf{else}\ J) = {\rm UsedInt}^*\,{\rm Loc}(I) \cup {\rm UsedInt}^*\,{\rm Loc}(J)$ and ${\rm UsedInt}^*\,{\rm Loc}(\mathbf{if}\ a > 0\ \mathbf{then}\ I\ \mathbf{else}\ J)$ $= {\rm UsedInt}^*\,{\rm Loc}(I) \cup {\rm UsedInt}^*\,{\rm Loc}(J)$.

(36) For every macro instruction I and for every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds ${\rm UsedInt}^*\,{\rm Loc}({\rm Directed}(I, l)) = {\rm UsedInt}^*\,{\rm Loc}(I)$.

(37) For every integer location a and for every macro instruction I holds ${\rm UsedInt}^*\,{\rm Loc}({\rm Times}(a, I)) = {\rm UsedInt}^*\,{\rm Loc}(I)$.

Let f be a finite sequence location and let t be a finite sequence of elements of $\mathbb{Z}$. Then $f \longmapsto t$ is a finite partial state of $\mathbf{SCM}_{\rm FSA}$.

One can prove the following propositions:

(38) Every finite sequence of elements of $\mathbb{Z}$ is a finite sequence of elements of $\mathbb{R}$.

(39) Let t be a finite sequence of elements of $\mathbb{Z}$. Then there exists a finite sequence u of elements of $\mathbb{R}$ such that t and u are fiberwise equipotent and u is a finite sequence of elements of $\mathbb{Z}$ and non-increasing.

(40) ${\rm dom}(({\rm intloc}(0) \longmapsto 1) {+\cdot}\ {\rm Start\text{-}At}({\rm insloc}(0))) = \{{\rm intloc}(0), \mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}\}$.

(41) For every macro instruction I holds $\mathrm{dom}\,\mathrm{Initialized}(I) = \mathrm{dom}\,I \cup \{\mathrm{intloc}(0), \mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}\}$.

(42) Let w be a finite sequence of elements of $\mathbb{Z}$, f be a finite sequence location, and I be a macro instruction. Then $\mathrm{dom}(\mathrm{Initialized}(I)+\cdot(f\dot{\longmapsto}w)) = \mathrm{dom}\,I \cup \{\mathrm{intloc}(0), \mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}, f\}$.

(43) For every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds $\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}} \neq l$.

(44) For every integer location a and for every macro instruction I holds $\mathrm{card}\,\mathrm{Times}(a, I) = \mathrm{card}\,I + 12$.

(45) For all instructions i_2, i_3, i_4 of $\mathbf{SCM}_{\rm FSA}$ holds $\mathrm{card}(i_2;i_3;i_4) = 6$, where $i_2 = b_4{:=}b_3$, $b_4 = \mathrm{intloc}(3+1)$, $b_3 = \mathrm{intloc}(2+1)$, $i_3 = \mathrm{SubFrom}(b_3, a_0)$, $a_0 = \mathrm{intloc}(0)$, $i_4 = b_5{:=}{f_0}_{b_3}$, $b_5 = \mathrm{intloc}(4+1)$, and $f_0 = \mathrm{fsloc}(0)$.

(46) Let t be a finite sequence of elements of $\mathbb{Z}$, f be a finite sequence location, and I be a macro instruction. Then $\mathrm{dom}\,\mathrm{Initialized}(I) \cap \mathrm{dom}(f\dot{\longmapsto}t) = \emptyset$.

(47) Let w be a finite sequence of elements of $\mathbb{Z}$, f be a finite sequence location, and I be a macro instruction. Then $\mathrm{Initialized}(I)+\cdot(f\dot{\longmapsto}w)$ starts at $\mathrm{insloc}(0)$.

(48) Let I, J be macro instructions, k be a natural number, and i be an instruction of $\mathbf{SCM}_{\rm FSA}$. If $k < \mathrm{card}\,J$ and $i = J(\mathrm{insloc}(k))$, then $(I;J)(\mathrm{insloc}(\mathrm{card}\,I + k)) = \mathrm{IncAddr}(i, \mathrm{card}\,I)$.

(49) Suppose that

(i) $i_1 = a{:=}b$, or
(ii) $i_1 = \mathrm{AddTo}(a, b)$, or
(iii) $i_1 = \mathrm{SubFrom}(a, b)$, or
(iv) $i_1 = \mathrm{MultBy}(a, b)$, or
(v) $i_1 = \mathrm{Divide}(a, b)$, or
(vi) $i_1 = \mathrm{goto}\ l_1$, or
(vii) $i_1 = \textbf{if } a = 0 \textbf{ goto } l_1$, or
(viii) $i_1 = \textbf{if } a > 0 \textbf{ goto } l_1$, or
(ix) $i_1 = b{:=}f_a$, or
(x) $i_1 = f_a{:=}b$, or
(xi) $i_1 = a{:=}\mathrm{len}f$, or
(xii) $i_1 = f{:=}\langle\underbrace{0, \ldots, 0}_{a}\rangle$.

Then $i_1 \neq \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$.

(50) Let I, J be macro instructions, k be a natural number, and i be an instruction of $\mathbf{SCM}_{\rm FSA}$. Suppose for every natural number n holds $\mathrm{IncAddr}(i, n) = i$ and $i \neq \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$ and $k = \mathrm{card}\,I$. Then $(I;i;J)(\mathrm{insloc}(k)) = i$ and $(I;i;J)(\mathrm{insloc}(k+1)) = \mathrm{goto}\ \mathrm{insloc}(\mathrm{card}\,I + 2)$.

(51) Let I, J be macro instructions and k be a natural number. If $k = \mathrm{card}\,I$, then $(I;(a{:=}b);J)(\mathrm{insloc}(k)) = a{:=}b$ and $(I;(a{:=}b);J)(\mathrm{insloc}(k+1)) =$

goto insloc(card $I+2$).

(52) Let I, J be macro instructions and k be a natural number. If $k = \operatorname{card} I$, then $(I;(a{:=}\operatorname{len} f);J)(\operatorname{insloc}(k)) = a{:=}\operatorname{len} f$ and $(I;(a{:=}\operatorname{len} f);J)(\operatorname{insloc}(k+1)) =$ goto insloc(card $I+2$).

(53) Let w be a finite sequence of elements of $\mathbb{Z}$, f be a finite sequence location, s be a state of $\mathbf{SCM}_{\rm FSA}$, and I be a macro instruction. If $\operatorname{Initialized}(I){+\cdot}(f{\dot{\longmapsto}}w) \subseteq s$, then $I \subseteq s$.

(54) Let w be a finite sequence of elements of $\mathbb{Z}$, f be a finite sequence location, s be a state of $\mathbf{SCM}_{\rm FSA}$, and I be a macro instruction. If $\operatorname{Initialized}(I){+\cdot}(f{\dot{\longmapsto}}w) \subseteq s$, then $s(f) = w$ and $s(\operatorname{intloc}(0)) = 1$.

(55) For every finite sequence location f and for every integer location a and for every state s of $\mathbf{SCM}_{\rm FSA}$ holds $\{a, \mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}, f\} \subseteq \operatorname{dom} s$.

(56) For every macro instruction p and for every state s of $\mathbf{SCM}_{\rm FSA}$ holds $\operatorname{UsedInt}^{*}\operatorname{Loc}(p) \cup \operatorname{UsedIntLoc}(p) \subseteq \operatorname{dom} s$.

(57) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a macro instruction, and f be a finite sequence location. Then $(\operatorname{Result}(s{+\cdot}\operatorname{Initialized}(I)))(f) = (\operatorname{IExec}(I,s))(f)$.

2. The Program Code for Buble Sort

Let f be a finite sequence location. The functor bubble-sort(f) yields a macro instruction and is defined as follows:

```
(Def. 1) bubble-sort($f$) = $i_5$;
    ($a_1$:=len$f$);
    Times($a_1$,
    ($a_2$:=$a_1$);
    SubFrom($a_2, a_0$);
    ($a_3$:=len$f$);
    Times($a_2$,
    ($a_4$:=$a_3$);
    SubFrom($a_3, a_0$);
    ($a_5$:=$f_{a_3}$);
    ($a_6$:=$f_{a_4}$);
    SubFrom($a_6, a_5$);
    (**if** $a_6 > 0$ **then** ($a_6$:=$f_{a_4}$);($f_{a_3}$:=$a_6$);($f_{a_4}$:=$a_5$) **else** (Stop$_{\mathrm{SCM}_{\mathrm{FSA}}}$)))),
    where $i_5 = (a_2{:=}a_0);(a_3{:=}a_0);(a_4{:=}a_0);(a_5{:=}a_0);(a_6{:=}a_0)$,
    $a_2$ = intloc(2), $a_0$ = intloc(0), $a_3$ = intloc(3), $a_4$ = intloc(4), $a_5$ =
    intloc(5), $a_6$ = intloc(6), and $a_1$ = intloc(1).
```

The macro instruction the bubble sort algorithm is defined by:

(Def. 2) The bubble sort algorithm = bubble-sort(fsloc(0)).

The following propositions are true:

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

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

3. Defining Relationship Between the Input and Output of Sorting Algorithms

The partial function Sorting-Function from FinPartSt($\mathbf{SCM}_{\mathrm{FSA}}$) to FinPartSt($\mathbf{SCM}_{\mathrm{FSA}}$) is defined by the condition (Def. 3).

(Def. 3) Let p, q be finite partial states of $\mathbf{SCM}_{\mathrm{FSA}}$. Then $\langle p, q\rangle \in$ Sorting-Function if and only if there exists a finite sequence t of elements of $\mathbb{Z}$ and there exists a finite sequence u of elements of $\mathbb{R}$ such that t and u are fiberwise equipotent and u is a finite sequence of elements of $\mathbb{Z}$ and non-increasing and $p =$ fsloc(0)$\dot{\longmapsto} t$ and $q =$ fsloc(0)$\dot{\longmapsto} u$.

We now state two propositions:

(60) For every set p holds $p \in$ dom Sorting-Function iff there exists a finite sequence t of elements of $\mathbb{Z}$ such that $p =$ fsloc(0)$\dot{\longmapsto} t$.

(61) Let t be a finite sequence of elements of $\mathbb{Z}$. 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) (Sorting-Function)(fsloc(0)$\dot{\longmapsto} t$) $=$ fsloc(0)$\dot{\longmapsto} u$.

4. The Basic Property of Buble Sort

Next we state several propositions:

(62) For every finite sequence location f holds card bubble-sort(f) $= 63$.

(63) For every finite sequence location f and for every natural number k such that $k < 63$ holds insloc(k) $\in$ dom bubble-sort(f).

(64) bubble-sort(fsloc(0)) is keepInt0 1 and InitHalting.

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

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

(ii) for all natural numbers i, j such that $i \geqslant 1$ and $j \leqslant \operatorname{len} s(f_0)$ and $i < j$ and for all integers x_1, x_2 such that $x_1 =$ (IExec(bubble-sort(f_0), s))(f_0)(i) and $x_2 =$ (IExec(bubble-sort(f_0), s))(f_0)(j) holds $x_1 \geqslant x_2$, where $f_0 = \mathrm{fsloc}(0)$.

(66) Let i be a natural number, s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, and w be a finite sequence of elements of $\mathbb{Z}$. Suppose Initialized(the bubble sort algorithm) $+\cdot(\mathrm{fsloc}(0) \mapsto w) \subseteq s$. Then $\mathbf{IC}_{(\mathrm{Computation}(s))(i)} \in \mathrm{dom}$ (the bubble sort algorithm).

(67) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$ and t be a finite sequence of elements of $\mathbb{Z}$. Suppose Initialized(the bubble sort algorithm) $+\cdot(\mathrm{fsloc}(0) \mapsto t) \subseteq s$. 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) $(\mathrm{Result}(s))(\mathrm{fsloc}(0)) = u$.

5. The Correctness and Autonomousness of Buble Sort Algorithm

We now state two propositions:

(68) For every finite sequence w of elements of $\mathbb{Z}$ holds Initialized(the bubble sort algorithm) $+\cdot(\mathrm{fsloc}(0) \mapsto w)$ is autonomic.

(69) Initialized(the bubble sort algorithm) computes Sorting-Function.

References

[1] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part I. *Formalized Mathematics*, 6(**1**):65–72, 1997.

[2] Noriko Asamoto. Conditional branch macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):73–80, 1997.

[3] Noriko Asamoto. Constant assignment macro instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II. *Formalized Mathematics*, 6(**1**):59–63, 1997.

[4] Noriko Asamoto. The `loop` and `Times` macroinstruction for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**4**):483–497, 1997.

[5] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Formalized Mathematics*, 6(**1**):41–47, 1997.

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

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

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

[9] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.

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

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

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

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

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

[15] 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.
[16] Jing-Chao Chen and Yatsuka Nakamura. Initialization halting concepts and their basic properties of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 7(**1**):139–151, 1998.
[17] Jarosław Kotowicz. Functions and finite sequences of real numbers. *Formalized Mathematics*, 3(**2**):275–278, 1992.
[18] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[19] Jan Popiołek. Some properties of functions modul and signum. *Formalized Mathematics*, 1(**2**):263–264, 1990.
[20] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 6(**1**):29–36, 1997.
[21] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[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 Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[25] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 5(**4**):571–576, 1996.
[26] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
[27] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.
[28] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[29] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[30] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 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 June 17, 1998