JOURNAL OF FORMALIZED MATHEMATICS
Volume 12, Released 2000, Published 2003
Inst. of Computer Science, Univ. of Białystok

Insert Sort on SCMPDS[1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. The goal of this article is to examine the effectiveness of "for-loop" and "while-loop" statements on SCMPDS by insert sort. In this article, first of all, we present an approach to compute the execution result of "for-loop" program by "loop-invariant", based on Hoare's axioms for program verification. Secondly, we extend the fundamental properties of the finite sequence and complex instructions of SCMPDS. Finally, we prove the correctness of the insert sort program described in the article.

MML Identifier: `SCPISORT`.

WWW: `http://mizar.org/JFM/Vol12/scpisort.html`

The articles [20], [5], [6], [19], [2], [16], [18], [4], [12], [7], [1], [10], [8], [9], [11], [3], [14], [13], [17], and [15] provide the notation and terminology for this paper.

1. Preliminaries

In this paper n, p_0 are natural numbers.

Let f be a finite sequence of elements of $\mathbb{Z}$, let s be a state of SCMPDS, and let m be a natural number. We say that f is FinSequence on s, m if and only if:

(Def. 1) For every natural number i such that $1 \leq i$ and $i \leq \mathrm{len}\, f$ holds $f(i) = s(\mathrm{intpos}\, m + i)$.

One can prove the following four propositions:

(1) Let f be a finite sequence of elements of $\mathbb{Z}$ and m, n be natural numbers. If $m \geq n$, then f is non decreasing on m, n.

(2) Let s be a state of SCMPDS and n, m be natural numbers. Then there exists a finite sequence f of elements of $\mathbb{Z}$ such that $\mathrm{len}\, f = n$ and for every natural number i such that $1 \leq i$ and $i \leq \mathrm{len}\, f$ holds $f(i) = s(\mathrm{intpos}\, m + i)$.

(3) Let s be a state of SCMPDS and n, m be natural numbers. Then there exists a finite sequence f of elements of $\mathbb{Z}$ such that $\mathrm{len}\, f = n$ and f is FinSequence on s, m.

(4) Let f, g be finite sequences of elements of $\mathbb{Z}$ and m, n be natural numbers. Suppose that $1 \leq n$ and $n \leq \mathrm{len}\, f$ and $1 \leq m$ and $m \leq \mathrm{len}\, f$ and $\mathrm{len}\, f = \mathrm{len}\, g$ and $f(m) = g(n)$ and $f(n) = g(m)$ and for every natural number k such that $k \neq m$ and $k \neq n$ and $1 \leq k$ and $k \leq \mathrm{len}\, f$ holds $f(k) = g(k)$. Then f and g are fiberwise equipotent.

We now state several propositions:

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

(5) For all states s_1, s_2 of SCMPDS such that for every Int position a holds $s_1(a) = s_2(a)$ holds Dstate s_1 = Dstate s_2.

(6) Let s be a state of SCMPDS, I be a No-StopCode Program-block, and j be a parahalting shiftable instruction of SCMPDS. Suppose I is closed on s and halting on s. Then I; j is closed on s and I; j is halting on s.

(7) Let s be a state of SCMPDS, I be a No-StopCode Program-block, J be a shiftable parahalting Program-block, and a be an Int position. If I is closed on s and halting on s, then $(\mathrm{IExec}(I;J,s))(a) = (\mathrm{IExec}(J,\mathrm{IExec}(I,s)))(a)$.

(8) Let s be a state of SCMPDS, I be a No-StopCode parahalting Program-block, J be a shiftable Program-block, and a be an Int position. If J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$, then $(\mathrm{IExec}(I;J,s))(a) = (\mathrm{IExec}(J,\mathrm{IExec}(I,s)))(a)$.

(9) Let s be a state of SCMPDS, I be a Program-block, and J be a shiftable parahalting Program-block. Suppose I is closed on s and halting on s. Then I; J is closed on s and I; J is halting on s.

(10) Let s be a state of SCMPDS, I be a parahalting Program-block, and J be a shiftable Program-block. Suppose J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$. Then I; J is closed on s and I; J is halting on s.

(11) Let s be a state of SCMPDS, I be a Program-block, and j be a parahalting shiftable instruction of SCMPDS. Suppose I is closed on s and halting on s. Then I; j is closed on s and I; j is halting on s.

2. Computing the Execution Result of For-Loop Program by Loop-Invariant

In this article we present several logical schemes. The scheme *ForDownHalt* deals with a state $\mathcal{A}$ of SCMPDS, a No-StopCode shiftable Program-block $\mathcal{B}$, an Int position $\mathcal{C}$, an integer $\mathcal{D}$, a natural number $\mathcal{E}$, and a unary predicate $\mathcal{P}$, and states that:

$\mathcal{P}[\mathcal{A}]$ or not $\mathcal{P}[\mathcal{A}]$ but for-down$(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B})$ is closed on $\mathcal{A}$ but for-down$(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B})$ is halting on $\mathcal{A}$

provided the parameters meet the following requirements:

- $\mathcal{E} > 0$,
- $\mathcal{P}[\mathrm{Dstate}\,\mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\mathrm{Dstate}\,t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) > 0$. Then $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t))(\mathcal{C}) = t(\mathcal{C})$ and $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t))(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) = t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) - \mathcal{E}$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{P}[\mathrm{Dstate}\,\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t)]$.

The scheme *ForDownExec* deals with a state $\mathcal{A}$ of SCMPDS, a No-StopCode shiftable Program-block $\mathcal{B}$, an Int position $\mathcal{C}$, an integer $\mathcal{D}$, a natural number $\mathcal{E}$, and a unary predicate $\mathcal{P}$, and states that:

$\mathcal{P}[\mathcal{A}]$ or not $\mathcal{P}[\mathcal{A}]$ but $\mathrm{IExec}(\text{for-down}(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B}),\mathcal{A}) = \mathrm{IExec}(\text{for-down}(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B}),\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),\mathcal{A}))$

provided the following requirements are met:

- $\mathcal{E} > 0$,
- $\mathcal{A}(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) > 0$,
- $\mathcal{P}[\mathrm{Dstate}\,\mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\mathrm{Dstate}\,t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) > 0$. Then $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t))(\mathcal{C}) = t(\mathcal{C})$ and $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t))(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) = t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) - \mathcal{E}$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{P}[\mathrm{Dstate}\,\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C},\mathcal{D},-\mathcal{E}),t)]$.

The scheme *ForDownEnd* deals with a state $\mathcal{A}$ of SCMPDS, a No-StopCode shiftable Program-block $\mathcal{B}$, an Int position $\mathcal{C}$, an integer $\mathcal{D}$, a natural number $\mathcal{E}$, and a unary predicate $\mathcal{P}$, and states that:

$\mathcal{P}[\mathcal{A}]$ or not $\mathcal{P}[\mathcal{A}]$ but $(\mathrm{IExec}(\text{for-down}(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B}),\mathcal{A}))(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) \leq 0$ but $\mathcal{P}[\mathrm{Dstate}\,\mathrm{IExec}(\text{for-down}(\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{B}),\mathcal{A})]$

provided the following requirements are met:

- $\mathcal{E} > 0$,

- $\mathcal{P}$[Dstate $\mathcal{A}$], and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}$[Dstate t] and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) > 0$. Then $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C}, \mathcal{D}, -\mathcal{E}), t))(\mathcal{C}) = t(\mathcal{C})$ and $(\mathrm{IExec}(\mathcal{B};\ \mathrm{AddTo}(\mathcal{C}, \mathcal{D}, -\mathcal{E}), t))(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) = t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) - \mathcal{E}$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{P}$[Dstate IExec($\mathcal{B}$; AddTo($\mathcal{C}, \mathcal{D}, -\mathcal{E}), t)$].

One can prove the following propositions:

(12) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, x, y be Int positions, i, c be integers, and n be a natural number. Suppose that

(i) $n > 0$,

(ii) $s(x) \geq s(y) + c$, and

(iii) for every state t of SCMPDS such that $t(x) \geq t(y) + c$ and $t(a) = s(a)$ and $t(\mathrm{DataLoc}(s(a), i)) > 0$ holds $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(a) = t(a)$ and $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(\mathrm{DataLoc}(s(a), i)) = t(\mathrm{DataLoc}(s(a), i)) - n$ and I is closed on t and halting on t and $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(x) \geq (\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(y) + c$.

Then for-down(a, i, n, I) is closed on s and for-down(a, i, n, I) is halting on s.

(13) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, x, y be Int positions, i, c be integers, and n be a natural number. Suppose that

(i) $n > 0$,

(ii) $s(x) \geq s(y) + c$,

(iii) $s(\mathrm{DataLoc}(s(a), i)) > 0$, and

(iv) for every state t of SCMPDS such that $t(x) \geq t(y) + c$ and $t(a) = s(a)$ and $t(\mathrm{DataLoc}(s(a), i)) > 0$ holds $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(a) = t(a)$ and $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(\mathrm{DataLoc}(s(a), i)) = t(\mathrm{DataLoc}(s(a), i)) - n$ and I is closed on t and halting on t and $(\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(x) \geq (\mathrm{IExec}(I;\ \mathrm{AddTo}(a, i, -n), t))(y) + c$.

Then IExec(for-down$(a, i, n, I), s)$ = IExec(for-down(a, i, n, I), IExec(I; AddTo$(a, i, -n), s)$).

(14) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a be an Int position, i be an integer, and n be a natural number. Suppose that

(i) $s(\mathrm{DataLoc}(s(a), i)) > 0$,

(ii) $n > 0$,

(iii) $\mathrm{card}\, I > 0$,

(iv) $a \neq \mathrm{DataLoc}(s(a), i)$, and

(v) for every state t of SCMPDS such that $t(a) = s(a)$ holds $(\mathrm{IExec}(I, t))(a) = t(a)$ and $(\mathrm{IExec}(I, t))(\mathrm{DataLoc}(s(a), i)) = t(\mathrm{DataLoc}(s(a), i))$ and I is closed on t and halting on t.

Then for-down(a, i, n, I) is closed on s and for-down(a, i, n, I) is halting on s.

3. A Program for Insert Sort

Let n, p_0 be natural numbers. The functor insert-sort(n, p_0) yields a Program-block and is defined by the condition (Def. 2).

(Def. 2) insert-sort(n, p_0) = (GBP:=0); ((GBP)$_1$:=0); ((GBP)$_2$:=$n-1$); ((GBP)$_3$:=p_0); for-down(GBP, 2, 1, AddTo(GBP, 3, 1); ((GBP, 4) := (GBP, 3)); AddTo(GBP, 1, 1); ((GBP, 6) := (GBP, 1)); while $>$ 0(GBP, 6, ((GBP, 5) := (intpos 4, -1)); SubFrom(GBP, 5, intpos 4, 0); (**if** GBP $>$ 5 **then** ((GBP, 5) := (intpos 4, -1)); ((intpos 4, -1) := (intpos 4, 0)); ((intpos 4, 0) := (GBP, 5)); AddTo(GBP, 4, -1); AddTo(GBP, 6, -1) **else** Load((GBP)$_6$:=0)))).

4. The Property of Insert Sort and Its Correctness

Next we state two propositions:

(15) card insert-sort$(n, p_0) = 23$.

(16) If $p_0 \geq 7$, then insert-sort(n, p_0) is parahalting.

One can prove the following two propositions:

(17) Let s be a state of SCMPDS, f, g be finite sequences of elements of $\mathbb{Z}$, and k_0, k be natural numbers. Suppose that $s(a_4) \geq 7 + s(a_6)$ and $s(\mathrm{GBP}) = 0$ and $k = s(a_6)$ and $k_0 = s(a_4) - s(a_6) - 1$ and f is FinSequence on s, k_0 and g is FinSequence on $\mathrm{IExec}(I_2, s)$, k_0 and $\mathrm{len}\, f = \mathrm{len}\, g$ and $\mathrm{len}\, f > k$ and f is non decreasing on 1, k. Then

(i) f and g are fiberwise equipotent,

(ii) g is non decreasing on 1, $k+1$,

(iii) for every natural number i such that $i > k+1$ and $i \leq \mathrm{len}\, f$ holds $f(i) = g(i)$, and

(iv) for every natural number i such that $1 \leq i$ and $i \leq k+1$ there exists a natural number j such that $1 \leq j$ and $j \leq k+1$ and $g(i) = f(j)$,

where $a_4 = \mathrm{intpos}\,4$, $a_6 = \mathrm{intpos}\,6$, $I_2 = W_1$, $W_1 = \mathrm{while} > 0(\mathrm{GBP}, 6, B_1)$, $B_1 = k_1; k_2; I_1$, $k_1 = (\mathrm{GBP}, 5) := (\mathrm{intpos}\,4, -1)$, $k_2 = \mathrm{SubFrom}(\mathrm{GBP}, 5, \mathrm{intpos}\,4, 0)$, $I_1 =$ **if** GBP > 5 **then** T_1 **else** F_1, $T_1 = k_3; k_4; k_5; k_6; k_7$, $k_3 = (\mathrm{GBP}, 5) := (\mathrm{intpos}\,4, -1)$, $k_4 = (\mathrm{intpos}\,4, -1) := (\mathrm{intpos}\,4, 0)$, $k_5 = (\mathrm{intpos}\,4, 0) := (\mathrm{GBP}, 5)$, $k_6 = \mathrm{AddTo}(\mathrm{GBP}, 4, -1)$, $k_7 = \mathrm{AddTo}(\mathrm{GBP}, 6, -1)$, and $F_1 = \mathrm{Load}((\mathrm{GBP})_6 {:=} 0)$.

(18) Let s be a state of SCMPDS, f, g be finite sequences of elements of $\mathbb{Z}$, and p_0, n be natural numbers. Suppose $p_0 \geq 6$ and $\mathrm{len}\, f = n$ and $\mathrm{len}\, g = n$ and f is FinSequence on s, p_0 and g is FinSequence on $\mathrm{IExec}(\text{insert-sort}(n, p_0 + 1), s)$, p_0. Then f and g are fiberwise equipotent and g is non decreasing on 1, n.

References

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

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

[3] 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`.

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

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

[6] 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`.

[7] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_3.html`.

[8] Jing-Chao Chen. Computation of two consecutive program blocks for SCMPDS. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_5.html`.

[9] Jing-Chao Chen. The construction and computation of conditional statements for SCMPDS. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_6.html`.

[10] Jing-Chao Chen. The construction and shiftability of program blocks for SCMPDS. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_4.html`.

[11] Jing-Chao Chen. Recursive Euclide algorithm. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmp_gcd.html`.

[12] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_2.html`.

[13] Jing-Chao Chen. The construction and computation of while-loop programs for SCMPDS. *Journal of Formalized Mathematics*, 12, 2000. `http://mizar.org/JFM/Vol12/scmpds_8.html`.

[14] Jing-Chao Chen and Piotr Rudnicki. The construction and computation of for-loop programs for SCMPDS. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_7.html`.

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

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

[17] Piotr Rudnicki. The for (going up) macro instruction. *Journal of Formalized Mathematics*, 10, 1998. http://mizar.org/JFM/Vol10/sfmastr3.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] Michał J. Trybulec. Integers. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/int_1.html.

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

Received June 14, 2000

Published January 2, 2004