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

The Construction and Computation of for-loop Programs for SCMPDS[1]

Jing-Chao Chen
Shanghai Jiaotong University

Piotr Rudnicki
University of Alberta

Summary. This article defines two for-loop statements for SCMPDS. One is called for-up, which corresponds to "for (i=x; i<0; i+=n) S" in C language. Another is called for-down, which corresponds to "for (i=x; i>0; i-=n) S". Here, we do not present their unconditional halting (called parahalting) property, because we have not found that there exists a useful for-loop statement with unconditional halting, and the proof of unconditional halting is much simpler than that of conditional halting. It is hard to formalize all halting conditions, but some cases can be formalized. We choose loop invariants as halting conditions to prove halting problem of for-up/down statements. When some variables (except the loop control variable) keep undestroyed on a set for the loop invariant, and the loop body is halting for this condition, the corresponding for-up/down is halting and computable under this condition. The computation of for-loop statements can be realized by evaluating its body. At the end of the article, we verify for-down statements by two examples for summing.

MML Identifier: SCMPDS_7.

WWW: http://mizar.org/JFM/Vol11/scmpds_7.html

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

1. PRELIMINARIES

For simplicity, we adopt the following rules: x denotes a set, n denotes a natural number, a denotes an Int position, i, j, k denote instructions of SCMPDS, s, s_1, s_2 denote states of SCMPDS, l_1, l denote instruction-locations of SCMPDS, and I, J, K denote Program-blocks.

The following propositions are true:

(1) For every state s of SCMPDS and for all natural numbers m, n such that $\mathbf{IC}_s = \mathrm{inspos}\, m$ holds $\mathrm{ICplusConst}(s, n-m) = \mathrm{inspos}\, n$.

(2) For all finite partial states P, Q of SCMPDS such that $P \subseteq Q$ holds $\mathrm{ProgramPart}(P) \subseteq \mathrm{ProgramPart}(Q)$.

(3) For all programmed finite partial states P, Q of SCMPDS and for every natural number k such that $P \subseteq Q$ holds $\mathrm{Shift}(P,k) \subseteq \mathrm{Shift}(Q,k)$.

(4) If $\mathbf{IC}_s = \mathrm{inspos}\, 0$, then $\mathrm{Initialized}(s) = s$.

(5) If $\mathbf{IC}_s = \mathrm{inspos}\, 0$, then $s +\cdot \mathrm{Initialized}(I) = s +\cdot I$.

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

© Association of Mizar Users

(6) $(\mathrm{Computation}(s))(n)\restriction$the instruction locations of SCMPDS $= s\restriction$the instruction locations of SCMPDS.

(7) Let s_1, s_2 be states of SCMPDS. Suppose $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and $s_1\restriction\text{Data-Loc}_{\mathrm{SCM}} = s_2\restriction\text{Data-Loc}_{\mathrm{SCM}}$ and $s_1\restriction$the instruction locations of SCMPDS $= s_2\restriction$the instruction locations of SCMPDS. Then $s_1 = s_2$.

(8) $l \in \mathrm{dom}\, I$ iff $l \in \mathrm{dom}\,\mathrm{Initialized}(I)$.

(9) If $x \in \mathrm{dom}\, I$, then $I(x) = (s{+}\cdot(I{+}\cdot\text{Start-At}(l)))(x)$.

(10) If $l_1 \in \mathrm{dom}\, I$, then $(s{+}\cdot\mathrm{Initialized}(I))(l_1) = I(l_1)$.

(11) $(s{+}\cdot(I{+}\cdot\text{Start-At}(l)))(a) = s(a)$.

(12) $(s{+}\cdot\text{Start-At}(l_1))(\mathbf{IC}_{\mathrm{SCMPDS}}) = l_1$.

(14)[1] $(I;\ i;\ j)(\mathrm{inspos}\,\mathrm{card}\, I) = i$.

(15) $(i;\ I;\ j);\ k = i;\ (I;\ j;\ k)$.

(16) $\mathrm{Shift}(J, \mathrm{card}\, I) \subseteq I;\ J;\ K$.

(17) $I \subseteq \mathrm{stop}\, I;\ J$.

(18) If $l_1 \in \mathrm{dom}\, I$, then $(\mathrm{Shift}(\mathrm{stop}\, I, n))(l_1 + n) = (\mathrm{Shift}(I, n))(l_1 + n)$.

(19) If $\mathrm{card}\, I > 0$, then $(\mathrm{Shift}(\mathrm{stop}\, I, n))(\mathrm{inspos}\, n) = (\mathrm{Shift}(I, n))(\mathrm{inspos}\, n)$.

(20) For every state s of SCMPDS and for every instruction i of SCMPDS such that $\mathrm{InsCode}(i) \in \{0, 4, 5, 6\}$ holds $\mathrm{Exec}(i, s)\restriction\text{Data-Loc}_{\mathrm{SCM}} = s\restriction\text{Data-Loc}_{\mathrm{SCM}}$.

(21) For all states s, s_3 of SCMPDS holds $(s{+}\cdot s_3\restriction$the instruction locations of SCMPDS$)\restriction\text{Data-Loc}_{\mathrm{SCM}} = s\restriction\text{Data-Loc}_{\mathrm{SCM}}$.

(22) For every instruction i of SCMPDS holds $\mathrm{rng}\,\mathrm{Load}(i) = \{i\}$.

(23) If $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and $s_1\restriction\text{Data-Loc}_{\mathrm{SCM}} = s_2\restriction\text{Data-Loc}_{\mathrm{SCM}}$, then $\mathbf{IC}_{\mathrm{Exec}(i,s_1)} = \mathbf{IC}_{\mathrm{Exec}(i,s_2)}$ and $\mathrm{Exec}(i, s_1)\restriction\text{Data-Loc}_{\mathrm{SCM}} = \mathrm{Exec}(i, s_2)\restriction\text{Data-Loc}_{\mathrm{SCM}}$.

(24) Let s_1, s_2 be states of SCMPDS and I be a Program-block. Suppose I is closed on s_1 and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_1$ and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_2$ and $s_1\restriction\text{Data-Loc}_{\mathrm{SCM}} = s_2\restriction\text{Data-Loc}_{\mathrm{SCM}}$. Let i be a natural number. Then $\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)} = \mathbf{IC}_{(\mathrm{Computation}(s_2))(i)}$ and $\mathrm{CurInstr}((\mathrm{Computation}(s_1))(i)) = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(i))$ and $(\mathrm{Computation}(s_1))(i)\restriction\text{Data-Loc}_{\mathrm{SCM}} = (\mathrm{Computation}(s_2))(i)\restriction\text{Data-Loc}_{\mathrm{SCM}}$.

(25) Let s_1, s_2 be states of SCMPDS and I be a Program-block. Suppose I is closed on s_1 and $s_1\restriction\text{Data-Loc}_{\mathrm{SCM}} = s_2\restriction\text{Data-Loc}_{\mathrm{SCM}}$. Let k be a natural number. Then $(\mathrm{Computation}(s_1{+}\cdot\mathrm{Initialized}(\mathrm{stop}\, I)))(k)$ and $(\mathrm{Computation}(s_2{+}\cdot\mathrm{Initialized}(\mathrm{stop}\, I)))(k)$ are equal outside the instruction locations of SCMPDS and $\mathrm{CurInstr}((\mathrm{Computation}(s_1{+}\cdot\mathrm{Initialized}(\mathrm{stop}\, I)))(k)) = \mathrm{CurInstr}((\mathrm{Computation}(s_2{+}\cdot\mathrm{Initialized}(\mathrm{stop}\, I)))(k))$.

(26) Let I be a Program-block. Suppose I is closed on s_1 and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_1$ and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_2$ and s_1 and s_2 are equal outside the instruction locations of SCMPDS. 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 SCMPDS and $\mathrm{CurInstr}((\mathrm{Computation}(s_1))(k)) = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(k))$.

(27) Let s_1, s_2 be states of SCMPDS and I be a Program-block. Suppose I is closed on s_1 and halting on s_1 and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_1$ and $\mathrm{Initialized}(\mathrm{stop}\, I) \subseteq s_2$ and $s_1\restriction\text{Data-Loc}_{\mathrm{SCM}} = s_2\restriction\text{Data-Loc}_{\mathrm{SCM}}$. Then $\mathrm{LifeSpan}(s_1) = \mathrm{LifeSpan}(s_2)$.

[1] The proposition (13) has been removed.

(28) Let I be a Program-block. Suppose that

(i) I is closed on s_1 and halting on s_1,

(ii) $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_1$,

(iii) $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_2$, and

(iv) s_1 and s_2 are equal outside the instruction locations of SCMPDS.

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 SCMPDS.

(29) Let s_1, s_2 be states of SCMPDS and I be a Program-block. Suppose I is closed on s_1 and halting on s_1 and $s_1 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = s_2 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$. Then $\mathrm{LifeSpan}(s_1 {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)) = \mathrm{LifeSpan}(s_2 {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I))$ and $\mathrm{Result}(s_1 {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I))$ and $\mathrm{Result}(s_2 {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I))$ are equal outside the instruction locations of SCMPDS.

(30) Let s_1, s_2 be states of SCMPDS and I be a Program-block. Suppose that

(i) I is closed on s_1 and halting on s_1,

(ii) $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_1$,

(iii) $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_2$, and

(iv) there exists a natural number k such that $(\mathrm{Computation}(s_1))(k)$ and s_2 are equal outside the instruction locations of SCMPDS.

Then $\mathrm{Result}(s_1)$ and $\mathrm{Result}(s_2)$ are equal outside the instruction locations of SCMPDS.

Let I be a Program-block. One can check that $\mathrm{Initialized}(I)$ is initial.
Next we state a number of propositions:

(31) Let s be a state of SCMPDS, I be a Program-block, and a be an Int position. If I is halting on s, then $(\mathrm{IExec}(I,s))(a) = (\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)))(\mathrm{LifeSpan}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)))(a)$.

(32) Let s be a state of SCMPDS, I be a parahalting Program-block, and a be an Int position. Then $(\mathrm{IExec}(I,s))(a) = (\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)))(\mathrm{LifeSpan}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)))(a)$.

(33) Let I be a Program-block and i be a natural number. If $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s$ and I is closed on s and halting on s and $i < \mathrm{LifeSpan}(s)$, then $\mathbf{IC}_{(\mathrm{Computation}(s))(i)} \in \mathrm{dom}\,I$.

(34) Let I be a shiftable Program-block. Suppose $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_1$ and I is closed on s_1 and halting on s_1. Let n be a natural number. Suppose $\mathrm{Shift}(I,n) \subseteq s_2$ and $\mathrm{card}\,I > 0$ and $\mathbf{IC}_{(s_2)} = \mathrm{inspos}\,n$ and $s_1 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = s_2 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$. Let i be a natural number. If $i < \mathrm{LifeSpan}(s_1)$, then $\mathbf{IC}_{(\mathrm{Computation}(s_1))(i)} + n = \mathbf{IC}_{(\mathrm{Computation}(s_2))(i)}$ and $\mathrm{CurInstr}((\mathrm{Computation}(s_1))(i)) = \mathrm{CurInstr}((\mathrm{Computation}(s_2))(i))$ and $(\mathrm{Computation}(s_1))(i) \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = (\mathrm{Computation}(s_2))(i) \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$.

(35) For every No-StopCode Program-block I such that $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s$ and I is halting on s and $\mathrm{card}\,I > 0$ holds $\mathrm{LifeSpan}(s) > 0$.

(36) Let I be a No-StopCode shiftable Program-block. Suppose $\mathrm{Initialized}(\mathrm{stop}\,I) \subseteq s_1$ and I is closed on s_1 and halting on s_1. Let n be a natural number. Suppose $\mathrm{Shift}(I,n) \subseteq s_2$ and $\mathrm{card}\,I > 0$ and $\mathbf{IC}_{(s_2)} = \mathrm{inspos}\,n$ and $s_1 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = s_2 \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$. Then $\mathbf{IC}_{(\mathrm{Computation}(s_2))(\mathrm{LifeSpan}(s_1))} = \mathrm{inspos}\,\mathrm{card}\,I + n$ and $(\mathrm{Computation}(s_1))(\mathrm{LifeSpan}(s_1)) \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = (\mathrm{Computation}(s_2))(\mathrm{LifeSpan}(s_1)) \restriction \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$.

(37) Let s be a state of SCMPDS, I be a Program-block, and n be a natural number. If $\mathbf{IC}_{(\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(I)))(n)} = \mathrm{inspos}\,0$, then $(\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(I)))(n) {+\cdot}\, \mathrm{Initialized}(I) = (\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(I)))(n)$.

(38) Let I be a Program-block, J be a Program-block, and k be a natural number. Suppose I is closed on s and halting on s and $k \leq \mathrm{LifeSpan}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I))$. Then $(\mathrm{Computation}(s {+\cdot}\, \mathrm{Initialized}(\mathrm{stop}\,I)))(k)$ and $(\mathrm{Computation}(s {+\cdot}\, ((I;J) {+\cdot}\, \mathrm{Start\text{-}At}(\mathrm{inspos}\,0))))(k)$ are equal outside the instruction locations of SCMPDS.

(39) Let I, J be Program-blocks and k be a natural number. Suppose $I \subseteq J$ and I is closed on s and halting on s and $k \leq \mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I))$. Then $(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(J)))(k)$ and $(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))(k)$ are equal outside the instruction locations of SCMPDS.

(40) Let I, J be Program-blocks and k be a natural number. Suppose $k \leq \mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I))$ and $I \subseteq J$ and I is closed on s and halting on s. Then $\mathbf{IC}_{(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(J)))(k)} \in \mathrm{dom}\,\mathrm{stop}\,I$.

(41) Let I, J be Program-blocks. Suppose $I \subseteq J$ and I is closed on s and halting on s. Then $\mathrm{CurInstr}((\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(J)))(\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))) = \mathbf{halt}_{\mathrm{SCMPDS}}$ or $\mathbf{IC}_{(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(J)))(\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))} = \mathrm{inspos}\,\mathrm{card}\,I$.

(42) Let I, J be Program-blocks. Suppose I is halting on s and J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$. Then J is closed on $(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))(\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))$ and halting on $(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))(\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))$.

(43) Let I be a Program-block and J be a shiftable Program-block. Suppose I is closed on s and halting on s and 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.

(44) Let I be a No-StopCode Program-block and J be a Program-block. If $I \subseteq J$ and I is closed on s and halting on s, then $\mathbf{IC}_{(\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(J)))(\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))} = \mathrm{inspos}\,\mathrm{card}\,I$.

(45) Let I be a Program-block, s be a state of SCMPDS, and k be a natural number. If I is halting on s and $k < \mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I))$, then $\mathrm{CurInstr}((\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)))(k)) \neq \mathbf{halt}_{\mathrm{SCMPDS}}$.

(46) Let I, J be Program-blocks, s be a state of SCMPDS, and k be a natural number. Suppose I is closed on s and halting on s and $k < \mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I))$. Then $\mathrm{CurInstr}((\mathrm{Computation}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I;J)))(k)) \neq \mathbf{halt}_{\mathrm{SCMPDS}}$.

(47) Let I be a No-StopCode Program-block and J be a shiftable Program-block. Suppose I is closed on s and halting on s and J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$. Then $\mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I;J)) = \mathrm{LifeSpan}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I)) + \mathrm{LifeSpan}(\mathrm{Result}(s+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,I))+\cdot\,\mathrm{Initialized}(\mathrm{stop}\,J))$.

(48) Let I be a No-StopCode Program-block and J be a shiftable Program-block. Suppose I is closed on s and halting on s and J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$. Then $\mathrm{IExec}(I;J,s) = \mathrm{IExec}(J,\mathrm{IExec}(I,s))+\cdot\,\mathrm{Start\text{-}At}(\mathbf{IC}_{\mathrm{IExec}(J,\mathrm{IExec}(I,s))} + \mathrm{card}\,I)$.

(49) Let I be a No-StopCode Program-block and J be a shiftable Program-block. Suppose I is closed on s and halting on s and J is closed on $\mathrm{IExec}(I,s)$ and halting on $\mathrm{IExec}(I,s)$. Then $(\mathrm{IExec}(I;J,s))(a) = (\mathrm{IExec}(J,\mathrm{IExec}(I,s)))(a)$.

(50) Let I be a No-StopCode Program-block and j be a parahalting shiftable instruction of SCMPDS. If I is closed on s and halting on s, then $(\mathrm{IExec}(I;j,s))(a) = (\mathrm{Exec}(j,\mathrm{IExec}(I,s)))(a)$.

2. The Construction of for-up loop Program

Let a be an Int position, let i be an integer, let n be a natural number, and let I be a Program-block. The functor for-up(a,i,n,I) yields a Program-block and is defined by:

(Def. 1) $\text{for-up}(a,i,n,I) = ((a,i) >= 0\_\mathrm{goto}\ \mathrm{card}\,I+3);\ I;\ \mathrm{AddTo}(a,i,n);\ \mathrm{goto}\ (-(\mathrm{card}\,I+2))$.

3. The Computation of for-up loop Program

Next we state several propositions:

(51) Let a be an Int position, i be an integer, n be a natural number, and I be a Program-block. Then $\mathrm{card}\,\text{for-up}(a,i,n,I) = \mathrm{card}\,I+3$.

(52) Let a be an Int position, i be an integer, n, m be natural numbers, and I be a Program-block. Then $m < \mathrm{card}\,I+3$ if and only if $\mathrm{inspos}\,m \in \mathrm{dom}\,\text{for-up}(a,i,n,I)$.

(53) Let a be an Int position, i be an integer, n be a natural number, and I be a Program-block. Then $(\text{for-up}(a,i,n,I))(\mathrm{inspos}\,0) = (a,i) >= 0\_\mathrm{goto}\,\mathrm{card}\,I+3$ and $(\text{for-up}(a,i,n,I))(\mathrm{inspos}\,\mathrm{card}\,I+1) = \mathrm{AddTo}(a,i,n)$ and $(\text{for-up}(a,i,n,I))(\mathrm{inspos}\,\mathrm{card}\,I+2) = \mathrm{goto}\,(-(\mathrm{card}\,I+2))$.

(54) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \geq 0$, then $\text{for-up}(a,i,n,I)$ is closed on s and $\text{for-up}(a,i,n,I)$ is halting on s.

(55) Let s be a state of SCMPDS, I be a Program-block, a, c be Int positions, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \geq 0$, then $\mathrm{IExec}(\text{for-up}(a,i,n,I),s) = s{+}\cdot\,\text{Start-At}(\mathrm{inspos}\,\mathrm{card}\,I+3)$.

(56) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \geq 0$, then $\mathbf{IC}_{\mathrm{IExec}(\text{for-up}(a,i,n,I),s)} = \mathrm{inspos}\,\mathrm{card}\,I+3$.

(57) Let s be a state of SCMPDS, I be a Program-block, a, b be Int positions, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \geq 0$, then $(\mathrm{IExec}(\text{for-up}(a,i,n,I),s))(b) = s(b)$.

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

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

(ii) $\mathrm{DataLoc}(s(a),i) \notin X$,

(iii) $n > 0$,

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

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

(vi) for every state t of SCMPDS such that for every Int position x such that $x \in X$ holds $t(x) = s(x)$ and $t(a) = s(a)$ 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 and for every Int position y such that $y \in X$ holds $(\mathrm{IExec}(I,t))(y) = t(y)$.

Then $\text{for-up}(a,i,n,I)$ is closed on s and $\text{for-up}(a,i,n,I)$ is halting on s.

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

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

(ii) $\mathrm{DataLoc}(s(a),i) \notin X$,

(iii) $n > 0$,

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

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

(vi) for every state t of SCMPDS such that for every Int position x such that $x \in X$ holds $t(x) = s(x)$ and $t(a) = s(a)$ 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 and for every Int position y such that $y \in X$ holds $(\mathrm{IExec}(I,t))(y) = t(y)$.

Then $\mathrm{IExec}(\text{for-up}(a,i,n,I),s) = \mathrm{IExec}(\text{for-up}(a,i,n,I),\mathrm{IExec}(I;\ \mathrm{AddTo}(a,i,n),s))$.

Let I be a shiftable Program-block, let a be an Int position, let i be an integer, and let n be a natural number. One can check that for-up(a,i,n,I) is shiftable.

Let I be a No-StopCode Program-block, let a be an Int position, let i be an integer, and let n be a natural number. Observe that for-up(a,i,n,I) is No-StopCode.

4. The Construction of for-down loop Program

Let a be an Int position, let i be an integer, let n be a natural number, and let I be a Program-block. The functor for-down(a,i,n,I) yielding a Program-block is defined as follows:

(Def. 2) for-down$(a,i,n,I) = ((a,i) <= 0_\mathrm{goto}\,\mathrm{card}\, I+3); I;\ \mathrm{AddTo}(a,i,-n);\ \mathrm{goto}\ (-(\mathrm{card}\, I+2))$.

5. The Computation of for-down loop Program

The following propositions are true:

(60) Let a be an Int position, i be an integer, n be a natural number, and I be a Program-block. Then $\mathrm{card}\,$for-down$(a,i,n,I) = \mathrm{card}\, I+3$.

(61) Let a be an Int position, i be an integer, n, m be natural numbers, and I be a Program-block. Then $m < \mathrm{card}\, I+3$ if and only if $\mathrm{inspos}\, m \in \mathrm{dom}\,$for-down$(a,i,n,I)$.

(62) Let a be an Int position, i be an integer, n be a natural number, and I be a Program-block. Then (for-down$(a,i,n,I))(\mathrm{inspos}\, 0) = (a,i) <= 0_\mathrm{goto}\,\mathrm{card}\, I+3$ and (for-down$(a,i,n,I))(\mathrm{inspos}\,\mathrm{card}\, I+1) = \mathrm{AddTo}(a,i,-n)$ and (for-down$(a,i,n,I))(\mathrm{inspos}\,\mathrm{card}\, I+2) = \mathrm{goto}\ (-(\mathrm{card}\, I+2))$.

(63) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \leq 0$, then for-down$(a,i,n,I)$ is closed on s and for-down(a,i,n,I) is halting on s.

(64) Let s be a state of SCMPDS, I be a Program-block, a, c be Int positions, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \leq 0$, then IExec(for-down$(a,i,n,I),s) = s+\cdot\,\mathrm{Start\text{-}At}(\mathrm{inspos}\,\mathrm{card}\, I+3)$.

(65) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \leq 0$, then $\mathbf{IC}_{\mathrm{IExec}(\text{for-down}(a,i,n,I),s)} = \mathrm{inspos}\,\mathrm{card}\, I+3$.

(66) Let s be a state of SCMPDS, I be a Program-block, a, b be Int positions, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a),i)) \leq 0$, then (IExec(for-down$(a,i,n,I),s))(b) = s(b)$.

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

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

(ii) $\mathrm{DataLoc}(s(a),i) \notin X$,

(iii) $n > 0$,

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

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

(vi) for every state t of SCMPDS such that for every Int position x such that $x \in X$ holds $t(x) = s(x)$ and $t(a) = s(a)$ 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 and for every Int position y such that $y \in X$ holds $(\mathrm{IExec}(I,t))(y) = t(y)$.

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

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

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

(ii) $\mathrm{DataLoc}(s(a),i) \notin X$,

(iii) $n > 0$,

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

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

(vi) for every state t of SCMPDS such that for every Int position x such that $x \in X$ holds $t(x) = s(x)$ and $t(a) = s(a)$ 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 and for every Int position y such that $y \in X$ holds $(\mathrm{IExec}(I,t))(y) = t(y)$.

Then $\mathrm{IExec}(\text{for-down}(a,i,n,I),s) = \mathrm{IExec}(\text{for-down}(a,i,n,I),\mathrm{IExec}(I;\ \mathrm{AddTo}(a,i,-n),s))$.

Let I be a shiftable Program-block, let a be an Int position, let i be an integer, and let n be a natural number. One can verify that for-down(a,i,n,I) is shiftable.

Let I be a No-StopCode Program-block, let a be an Int position, let i be an integer, and let n be a natural number. Note that for-down(a,i,n,I) is No-StopCode.

6. Two Examples for Summing

Let n be a natural number. The functor $\mathrm{sum}\, n$ yields a Program-block and is defined as follows:

(Def. 3) $\mathrm{sum}\, n = (\mathrm{GBP}{:=}0);\ ((\mathrm{GBP})_2{:=}n);\ ((\mathrm{GBP})_3{:=}0);\ \text{for-down}(\mathrm{GBP},2,1,\mathrm{Load}(\mathrm{AddTo}(\mathrm{GBP},3,1)))$.

We now state three propositions:

(69) For every state s of SCMPDS such that $s(\mathrm{GBP}) = 0$ holds for-down$(\mathrm{GBP},2,1,\mathrm{Load}(\mathrm{AddTo}(\mathrm{GBP},3,1)))$ is closed on s and for-down$(\mathrm{GBP},2,1,\mathrm{Load}(\mathrm{AddTo}(\mathrm{GBP},3,1)))$ is halting on s.

(70) Let s be a state of SCMPDS and n be a natural number. If $s(\mathrm{GBP}) = 0$ and $s(\mathrm{intpos}\,2) = n$ and $s(\mathrm{intpos}\,3) = 0$, then $(\mathrm{IExec}(\text{for-down}(\mathrm{GBP},2,1,\mathrm{Load}(\mathrm{AddTo}(\mathrm{GBP},3,1))),s))(\mathrm{intpos}\,3) = n$.

(71) For every state s of SCMPDS and for every natural number n holds $(\mathrm{IExec}(\mathrm{sum}\, n,s))(\mathrm{intpos}\,3) = n$.

Let s_4, c_1, r_1, p_1, p_2 be natural numbers. The functor $\mathrm{sum}(s_4,c_1,r_1,p_1,p_2)$ yields a Program-block and is defined as follows:

(Def. 4) $\mathrm{sum}(s_4,c_1,r_1,p_1,p_2) = ((\mathrm{intpos}\, s_4)_{r_1}{:=}0);\ (\mathrm{intpos}\, p_1{:=}p_2);\ \text{for-down}(\mathrm{intpos}\, s_4,c_1,1,\mathrm{AddTo}(\mathrm{intpos}\, s_4,r_1,\mathrm{intpos}\, p_2,0);\ \mathrm{AddTo}(\mathrm{intpos}\, p_1,0,1))$.

One can prove the following propositions:

(72) Let s be a state of SCMPDS and s_4, c_2, r_1, p_1, p_3 be natural numbers. Suppose $s(\mathrm{intpos}\, s_4) > s_4$ and $c_2 < r_1$ and $s(\mathrm{intpos}\, p_1) = p_3$ and $s(\mathrm{intpos}\, s_4) + r_1 < p_1$ and $p_1 < p_3$ and $p_3 < s(\mathrm{intpos}\, p_3)$. Then for-down$(\mathrm{intpos}\, s_4,c_2,1,\mathrm{AddTo}(\mathrm{intpos}\, s_4,r_1,\mathrm{intpos}\, p_3,0);\ \mathrm{AddTo}(\mathrm{intpos}\, p_1,0,1))$ is closed on s and for-down$(\mathrm{intpos}\, s_4,c_2,1,\mathrm{AddTo}(\mathrm{intpos}\, s_4,r_1,\mathrm{intpos}\, p_3,0);\ \mathrm{AddTo}(\mathrm{intpos}\, p_1,0,1))$ is halting on s.

(73) Let s be a state of SCMPDS, s_4, c_2, r_1, p_1, p_3 be natural numbers, and f be a finite sequence of elements of $\mathbb{N}$. Suppose that $s(\mathrm{intpos}\, s_4) > s_4$ and $c_2 < r_1$ and $s(\mathrm{intpos}\, p_1) = p_3$ and $s(\mathrm{intpos}\, s_4) + r_1 < p_1$ and $p_1 < p_3$ and $p_3 < s(\mathrm{intpos}\, p_3)$ and $s(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4),r_1)) = 0$ and $\mathrm{len}\, f = s(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4),c_2))$ and for every natural number k such that $k < \mathrm{len}\, f$ holds $f(k+1) = s(\mathrm{DataLoc}(s(\mathrm{intpos}\, p_3),k))$. Then $(\mathrm{IExec}(\text{for-down}(\mathrm{intpos}\, s_4,c_2,1,\mathrm{AddTo}(\mathrm{intpos}\, s_4,r_1,\mathrm{intpos}\, p_3,0);\ \mathrm{AddTo}(\mathrm{intpos}\, p_1,0,1)),s))(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4),r_1)) = \sum f$.

(74) Let s be a state of SCMPDS, s_4, c_2, r_1, p_1, p_3 be natural numbers, and f be a finite sequence of elements of $\mathbb{N}$. Suppose that $s(\mathrm{intpos}\, s_4) > s_4$ and $c_2 < r_1$ and $s(\mathrm{intpos}\, s_4) + r_1 < p_1$ and $p_1 < p_3$ and $p_3 < s(\mathrm{intpos}\, p_3)$ and $\mathrm{len}\, f = s(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4), c_2))$ and for every natural number k such that $k < \mathrm{len}\, f$ holds $f(k+1) = s(\mathrm{DataLoc}(s(\mathrm{intpos}\, p_3), k))$. Then $(\mathrm{IExec}(\mathrm{sum}(s_4, c_2, r_1, p_1, p_3), s))(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4), r_1)) = \sum f$.

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. Joining of decorated trees. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/trees_4.html`.

[4] 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`.

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

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

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

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

[9] 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`.

[10] 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`.

[11] 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`.

[12] 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`.

[13] 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`.

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

[15] 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`.

[16] Andrzej Kondracki. The Chinese Remainder Theorem. *Journal of Formalized Mathematics*, 9, 1997. `http://mizar.org/JFM/Vol9/wsierp_1.html`.

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

[18] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Journal of Formalized Mathematics*, 4, 1992. `http://mizar.org/JFM/Vol4/ami_2.html`.

[19] Jan Popiołek. Some properties of functions modul and signum. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/absvalue.html`.

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

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

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

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

[24] 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`.

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

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

Received December 27, 1999

Published January 2, 2004