FORMALIZED MATHEMATICS
Volume 9, Number 1, 2001
University 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.

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

1. Preliminaries

For simplicity, we adopt the following convention: x is a set, n is a natural number, a is a Int position, i, j, k are instructions of SCMPDS, s, s_1, s_2 are

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

ISSN 1426–2630

states of SCMPDS, l_1, l are instructions-locations of SCMPDS, and I, J, K are Program-block.

We now state a number of propositions:

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

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

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

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

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

(6) $(\operatorname{Computation}(s))(n)\restriction$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 $s_1\restriction\text{Data-Loc}_{\text{SCM}} = s_2\restriction\text{Data-Loc}_{\text{SCM}}$ and $s_1\restriction$the instruction locations of SCMPDS $= s_2\restriction$the instruction locations of SCMPDS. Then $s_1 = s_2$.

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

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

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

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

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

(13) $\operatorname{card}(I;i) = \operatorname{card} I + 1$.

(14) $(I;i;j)(\operatorname{inspos}\operatorname{card} I) = i$.

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

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

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

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

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

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

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

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

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

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

(26) Let I be a Program-block. Suppose that
- (i) I is closed on s_1 and halting on s_1,
- (ii) Initialized(stop I) $\subseteq$ s_1,
- (iii) Initialized(stop I) $\subseteq$ s_2, and
- (iv) s_1 and s_2 are equal outside the instruction locations of SCMPDS.

Let k be a natural number. Then (Computation(s_1))(k) and (Computation(s_2))(k) are equal outside the instruction locations of SCMPDS and CurInstr((Computation(s_1))(k)) = CurInstr((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 Initialized(stop I) $\subseteq$ s_1 and Initialized(stop I) $\subseteq$ s_2 and $s_1\restriction$Data-Loc$_{\mathrm{SCM}}$ = $s_2\restriction$Data-Loc$_{\mathrm{SCM}}$. Then LifeSpan(s_1) = LifeSpan(s_2).

(28) Let I be a Program-block. Suppose that
- (i) I is closed on s_1 and halting on s_1,
- (ii) Initialized(stop I) $\subseteq$ s_1,
- (iii) Initialized(stop I) $\subseteq$ s_2, and
- (iv) s_1 and s_2 are equal outside the instruction locations of SCMPDS.

Then LifeSpan(s_1) = LifeSpan(s_2) and Result(s_1) and 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 $s_1\restriction$Data-Loc$_{\mathrm{SCM}}$ = $s_2\restriction$Data-Loc$_{\mathrm{SCM}}$. Then LifeSpan(s_1+$\cdot$ Initialized(stop I)) = LifeSpan(s_2+$\cdot$ Initialized(stop I)) and Result(s_1+$\cdot$ Initialized(stop I)) and Result(s_2+$\cdot$ Initialized(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) Initialized(stop I) $\subseteq$ s_1,

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

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

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

Let I be a Program-block. One can check that Initialized(I) is initial.

The following propositions are true:

(31) Let s be a state of SCMPDS, I be a Program-block, and a be a Int position. If I is halting on s, then (IExec(I, s))(a) = (Computation(s+· Initialized(stop I)))(LifeSpan(s+· Initialized(stop I)))(a).

(32) Let s be a state of SCMPDS, I be a parahalting Program-block, and a be a Int position. Then (IExec(I, s))(a) = (Computation(s+· Initialized(stop I)))(LifeSpan(s+· Initialized(stop I)))(a).

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

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

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

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

(37) Let s be a state of SCMPDS, I be a Program-block, and n be a natural number. If $\mathbf{IC}_{(\text{Computation}(s\text{+·}\,\text{Initialized}(I)))(n)} =$ inspos 0, then (Computation(s+· Initialized(I)))(n)+· Initialized(I) = (Computation(s+· 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 $k \leqslant$ LifeSpan(s+· Initialized(stop I)). Then (Computation(s+· Initialized (stop I)))(k) and (Computation(s+·((I;J)+· Start-At(inspos 0))))(k) are

equal outside the instruction locations of SCMPDS.

(39) Let I, J be Program-block and k be a natural number. Suppose $I \subseteq J$ and I is closed on s and halting on s and $k \leqslant \mathrm{LifeSpan}(s+\cdot \mathrm{Initialized}(\mathrm{stop}\, I))$. 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-block and k be a natural number. Suppose $k \leqslant \mathrm{LifeSpan}(s+\cdot \mathrm{Initialized}(\mathrm{stop}\, I))$ 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-block. 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-block. 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-block, 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 a 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) yielding a Program-block is defined by:

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

3. The Computation of for-up loop Program

We now state several propositions:

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

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

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

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

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

(57) Let s be a state of SCMPDS, I be a Program-block, a, b be Int position, i be an integer, and n be a natural number. If $s(\mathrm{DataLoc}(s(a), i)) \geqslant 0$, 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 a 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-up(a, i, n, I) is closed on s and 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 a 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 a Int position, let i be an integer, and let n be a natural number. Observe that for-up(a, i, n, I) is shiftable.

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

4. The Construction of for-down loop Program

Let a be a 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$(a, i, n, I) = ((a, i) <= 0_goto \operatorname{card} I + 3); I;$ AddTo$(a, i, -n);$ goto $(-(\operatorname{card} I + 2))$.

5. The Computation of for-down loop Program

One can prove the following propositions:

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

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

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

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

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

(66) Let s be a state of SCMPDS, I be a Program-block, a, b be Int position, i be an integer, and n be a natural number. If $s($DataLoc$(s(a), i)) \leqslant 0$, 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 a Int position, i be an integer, n be a natural number, and X be a set. Suppose that

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

(68) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a be a 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 $\mathrm{IExec}(\mathrm{for-down}(a, i, n, I), s) = \mathrm{IExec}(\mathrm{for-down}(a, i, n, I), \mathrm{IExec}(I; \mathrm{AddTo}(a, i, -n), s))$.

Let I be a shiftable Program-block, let a be a Int position, let i be an integer, and let n be a natural number. Observe that $\mathrm{for-down}(a, i, n, I)$ is shiftable.

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

6. Two Examples for Summing

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

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

Next we state three propositions:

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

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

Next we state three 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 $\mathrm{for-down}(\mathrm{intpos}\, s_4, c_2, 1, \mathrm{AddTo}(\mathrm{intpos}\, s_4, r_1, \mathrm{intpos}\, p_3, 0);$ $\mathrm{AddTo}(\mathrm{intpos}\, p_1, 0, 1))$ is closed on s and $\mathrm{for-down}(\mathrm{intpos}\, s_4, c_2, 1,$ $\mathrm{AddTo}(\mathrm{intpos}\, s_4, r_1, \mathrm{intpos}\, p_3, 0); \mathrm{AddTo}(\mathrm{intpos}\, p_1, 0, 1))$ 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 $(\mathrm{IExec}(\mathrm{for-down}(\mathrm{intpos}\, s_4, c_2, 1, \mathrm{AddTo}(\mathrm{intpos}\, s_4, r_1, \mathrm{intpos}\, p_3, 0);$ $\mathrm{AddTo}(\mathrm{intpos}\, p_1, 0, 1)), s))(\mathrm{DataLoc}(s(\mathrm{intpos}\, s_4), r_1)) = \sum f$.

(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. *Formalized Mathematics*, 1(**2**):377–382, 1990.
[2] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.

[3] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.
[4] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[5] Grzegorz Bancerek and Piotr Rudnicki. Development of terminology for **scm**. *Formalized Mathematics*, 4(**1**):61–67, 1993.
[6] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.
[7] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[8] 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.
[9] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Formalized Mathematics*, 8(**1**):193–199, 1999.
[10] Jing-Chao Chen. Computation of two consecutive program blocks for SCMPDS. *Formalized Mathematics*, 8(**1**):211–217, 1999.
[11] Jing-Chao Chen. The construction and computation of conditional statements for SCMPDS. *Formalized Mathematics*, 8(**1**):219–234, 1999.
[12] Jing-Chao Chen. The construction and shiftability of program blocks for SCMPDS. *Formalized Mathematics*, 8(**1**):201–210, 1999.
[13] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Formalized Mathematics*, 8(**1**):183–191, 1999.
[14] Jing-Chao Chen. Recursive Euclide algorithm. *Formalized Mathematics*, 9(**1**):1–4, 2001.
[15] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[16] Andrzej Kondracki. The chinese remainder theorem. *Formalized Mathematics*, 6(**4**):573–577, 1997.
[17] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[18] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[19] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[20] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[21] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[22] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[23] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[24] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[25] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received December 27, 1999