FORMALIZED MATHEMATICS
Volume 9, Number 2, 2001
University of Białystok

Justifying the Correctness of the Fibonacci Sequence and the Euclide Algorithm by Loop-Invariant[1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. If a loop-invariant exists in a loop program, computing its result by loop-invariant is simpler and easier than computing its result by the inductive method. For this purpose, the article describes the premise and the final computation result of the program such as "while<0", "while>0", "while<>0" by loop-invariant. To test the effectiveness of the computation method given in this article, by using loop-invariant of the loop programs mentioned above, we justify the correctness of the following three examples: Summing n integers (used for testing "while>0"), Fibonacci sequence (used for testing "while<0"), Greatest Common Divisor, i.e. Euclide algorithm (used for testing "while<>0").

MML Identifier: `SCPINVAR`.

The notation and terminology used here have been introduced in the following papers: [18], [22], [19], [1], [3], [4], [6], [7], [24], [23], [2], [5], [16], [26], [27], [12], [8], [11], [9], [10], [13], [15], [14], [21], [25], [20], and [17].

1. Preliminaries

For simplicity, we adopt the following rules: m, n are natural numbers, i, j are instructions of SCMPDS, I is a Program-block, and a is an Int position.

One can prove the following propositions:

(1) For all natural numbers n, m, l such that $n \mid m$ and $n \mid l$ holds $n \mid m - l$.

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

ISSN 1426–2630

(2) $m \mid n$ iff $m \mid n$ **qua** integer.

(3) $\gcd(m,n) = \gcd(m, |n-m|)$.

(4) For all integers a, b such that $a \geqslant 0$ and $b \geqslant 0$ holds $a \gcd b = a \gcd b - a$.

(5) $(i;\ j;\ I)(\text{inspos}\,0) = i$ and $(i;\ j;\ I)(\text{inspos}\,1) = j$.

(6) Let a, b be Int positions. Then there exists a function f from $\prod$(the object kind of SCMPDS) into $\mathbb{N}$ such that for every state s of SCMPDS holds

(i) if $s(a) = s(b)$, then $f(s) = 0$, and

(ii) if $s(a) \neq s(b)$, then $f(s) = \max(|s(a)|, |s(b)|)$.

(7) There exists a function f from $\prod$(the object kind of SCMPDS) into $\mathbb{N}$ such that for every state s of SCMPDS holds

(i) if $s(a) \geqslant 0$, then $f(s) = 0$, and

(ii) if $s(a) < 0$, then $f(s) = -s(a)$.

2. Computing Directly the Result of "while<0" Program by Loop-Invariant

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

$\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{A})$ or $\mathcal{P}[\mathcal{A}]$ but $\mathcal{F}(\text{Dstate}\,\text{IExec}(\text{while} < 0(\mathcal{C},\mathcal{D},\mathcal{B}),\mathcal{A})) = 0$ but $\mathcal{P}[\text{Dstate}\,\text{IExec}(\text{while} < 0(\mathcal{C},\mathcal{D},\mathcal{B}),\mathcal{A})]$

provided the parameters satisfy the following conditions:

- $\text{card}\,\mathcal{B} > 0$,
- For every state t of SCMPDS such that $\mathcal{P}[\text{Dstate}\,t]$ holds $\mathcal{F}(\text{Dstate}\,t) = 0$ iff $t(\text{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) \geqslant 0$,
- $\mathcal{P}[\text{Dstate}\,\mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\text{Dstate}\,t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\text{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) < 0$. Then $(\text{IExec}(\mathcal{B},t))(\mathcal{C}) = t(\mathcal{C})$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{F}(\text{Dstate}\,\text{IExec}(\mathcal{B},t)) < \mathcal{F}(\text{Dstate}\,t)$ and $\mathcal{P}[\text{Dstate}\,\text{IExec}(\mathcal{B},t)]$.

3. An Example: Summing Directly n Integers by Loop-Invariant

Let n, p_0 be natural numbers. The functor $\text{sum}(n, p_0)$ yields a Program-block and is defined as follows:

(Def. 1) $\text{sum}(n,p_0) = (\text{GBP} := 0);\ (\text{intpos}\,1 := 0);\ (\text{intpos}\,2 := -n);\ (\text{intpos}\,3 := p_0 + 1);\ \text{while} < 0(\text{GBP}, 2, \text{AddTo}(\text{GBP}, 1, \text{intpos}\,3, 0);\ \text{AddTo}(\text{GBP}, 2, 1);\ \text{AddTo}(\text{GBP}, 3, 1))$.

We now state the proposition

(8) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, b, c be Int positions, n, i, p_0 be natural numbers, and f be a finite sequence of elements of $\mathbb{Z}$. Suppose that $\operatorname{card} I > 0$ and f is FinSequence on s, p_0 and $\operatorname{len} f = n$ and $s(b) = 0$ and $s(a) = 0$ and $s(\operatorname{intpos} i) = -n$ and $s(c) = p_0 + 1$ and for every state t of SCMPDS such that there exists a finite sequence g of elements of $\mathbb{Z}$ such that g is FinSequence on s, p_0 and $\operatorname{len} g = t(\operatorname{intpos} i) + n$ and $t(b) = \sum g$ and $t(c) = p_0 + 1 + \operatorname{len} g$ and $t(a) = 0$ and $t(\operatorname{intpos} i) < 0$ and for every natural number i such that $i > p_0$ holds $t(\operatorname{intpos} i) = s(\operatorname{intpos} i)$ holds $(\mathrm{IExec}(I, t))(a) = 0$ and I is closed on t and halting on t and $(\mathrm{IExec}(I, t))(\operatorname{intpos} i) = t(\operatorname{intpos} i) + 1$ and there exists a finite sequence g of elements of $\mathbb{Z}$ such that g is FinSequence on s, p_0 and $\operatorname{len} g = t(\operatorname{intpos} i) + n + 1$ and $(\mathrm{IExec}(I, t))(c) = p_0 + 1 + \operatorname{len} g$ and $(\mathrm{IExec}(I, t))(b) = \sum g$ and for every natural number i such that $i > p_0$ holds $(\mathrm{IExec}(I, t))(\operatorname{intpos} i) = s(\operatorname{intpos} i)$. Then $(\mathrm{IExec}(\text{while} < 0(a, i, I), s))(b) = \sum f$ and while $< 0(a, i, I)$ is closed on s and while $< 0(a, i, I)$ is halting on s.

One can prove the following proposition

(9) Let s be a state of SCMPDS, n, p_0 be natural numbers, and f be a finite sequence of elements of $\mathbb{Z}$. Suppose $p_0 \geqslant 3$ and f is FinSequence on s, p_0 and $\operatorname{len} f = n$. Then $(\mathrm{IExec}(\operatorname{sum}(n, p_0), s))(\operatorname{intpos} 1) = \sum f$ and $\operatorname{sum}(n, p_0)$ is parahalting.

4. Computing Directly the Result of "while>0" Program by Loop-Invariant

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

$\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{A})$ or $\mathcal{P}[\mathcal{A}]$ but $\mathcal{F}(\operatorname{Dstate} \mathrm{IExec}(\text{while} > 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathcal{A})) = 0$ but $\mathcal{P}[\operatorname{Dstate} \mathrm{IExec}(\text{while} > 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathcal{A})]$

provided the parameters meet the following requirements:

- $\operatorname{card} \mathcal{B} > 0$,
- For every state t of SCMPDS such that $\mathcal{P}[\operatorname{Dstate} t]$ holds $\mathcal{F}(\operatorname{Dstate} t) = 0$ iff $t(\operatorname{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \leqslant 0$,
- $\mathcal{P}[\operatorname{Dstate} \mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\operatorname{Dstate} t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\operatorname{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) > 0$. Then $(\mathrm{IExec}(\mathcal{B}, t))(\mathcal{C}) = t(\mathcal{C})$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{F}(\operatorname{Dstate} \mathrm{IExec}(\mathcal{B}, t)) < \mathcal{F}(\operatorname{Dstate} t)$ and $\mathcal{P}[\operatorname{Dstate} \mathrm{IExec}(\mathcal{B}, t)]$.

5. An Example: Computing Directly Fibonacci Sequence by Loop-Invariant

Let n be a natural number. The functor Fib-macro n yields a Program-block and is defined by:

(Def. 2) Fib-macro n = (GBP :=0); (intpos 1:=0); (intpos 2:=1); (intpos 3:=n); while $> 0(\mathrm{GBP}, 3, ((\mathrm{GBP}, 4) := (\mathrm{GBP}, 2));\ \mathrm{AddTo}(\mathrm{GBP}, 2, \mathrm{GBP}, 1);$ $((\mathrm{GBP}, 1) := (\mathrm{GBP}, 4));\ \mathrm{AddTo}(\mathrm{GBP}, 3, -1))$.

We now state the proposition

(10) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, f_0, f_1 be Int positions, and n, i be natural numbers. Suppose that

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

(ii) $s(a) = 0$,

(iii) $s(f_0) = 0$,

(iv) $s(f_1) = 1$,

(v) $s(\mathrm{intpos}\, i) = n$, and

(vi) for every state t of SCMPDS and for every natural number k such that $n = t(\mathrm{intpos}\, i) + k$ and $t(f_0) = \mathrm{Fib}(k)$ and $t(f_1) = \mathrm{Fib}(k+1)$ and $t(a) = 0$ and $t(\mathrm{intpos}\, i) > 0$ holds $(\mathrm{IExec}(I, t))(a) = 0$ and I is closed on t and halting on t and $(\mathrm{IExec}(I, t))(\mathrm{intpos}\, i) = t(\mathrm{intpos}\, i) - 1$ and $(\mathrm{IExec}(I, t))(f_0) = \mathrm{Fib}(k+1)$ and $(\mathrm{IExec}(I, t))(f_1) = \mathrm{Fib}(k+1+1)$.

Then $(\mathrm{IExec}(\mathrm{while} > 0(a, i, I), s))(f_0) = \mathrm{Fib}(n)$ and $(\mathrm{IExec}(\mathrm{while} > 0(a, i, I), s))(f_1) = \mathrm{Fib}(n+1)$ and while $> 0(a, i, I)$ is closed on s and while $> 0(a, i, I)$ is halting on s.

One can prove the following proposition

(11) For every state s of SCMPDS and for every natural number n holds $(\mathrm{IExec}(\text{Fib-macro}\, n, s))(\mathrm{intpos}\, 1) = \mathrm{Fib}(n)$ and $(\mathrm{IExec}(\text{Fib-macro}\, n, s))(\mathrm{intpos}\, 2) = \mathrm{Fib}(n+1)$ and Fib-macro n is parahalting.

6. The Construction of "while<>0" Loop Program

Let a be an Int position, let i be an integer, and let I be a Program-block. The functor while $<> 0(a, i, I)$ yields a Program-block and is defined as follows:

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

7. The Basic Property of "while<>0" Program

One can prove the following propositions:

(12) For every Int position a and for every integer i and for every Program-block I holds $\operatorname{card} \mathrm{while} <> 0(a, i, I) = \operatorname{card} I + 3$.

(13) Let a be an Int position, i be an integer, m be a natural number, and I be a Program-block. Then $m < \operatorname{card} I + 3$ if and only if $\mathrm{inspos}\, m \in \operatorname{dom} \mathrm{while} <> 0(a, i, I)$.

(14) For every Int position a and for every integer i and for every Program-block I holds $\mathrm{inspos}\, 0 \in \operatorname{dom} \mathrm{while} <> 0(a, i, I)$ and $\mathrm{inspos}\, 1 \in \operatorname{dom} \mathrm{while} <> 0(a, i, I)$.

(15) Let a be an Int position, i be an integer, and I be a Program-block. Then $(\mathrm{while} <> 0(a, i, I))(\mathrm{inspos}\, 0) = (a, i) <> 0\_\mathrm{goto} 2$ and $(\mathrm{while} <> 0(a, i, I))(\mathrm{inspos}\, 1) = \mathrm{goto}\, (\operatorname{card} I + 2)$ and $(\mathrm{while} <> 0(a, i, I))(\mathrm{inspos}\, \operatorname{card} I + 2) = \mathrm{goto}\, (-(\operatorname{card} I + 2))$.

(16) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, and i be an integer. If $s(\mathrm{DataLoc}(s(a), i)) = 0$, then $\mathrm{while} <> 0(a, i, I)$ is closed on s and $\mathrm{while} <> 0(a, i, I)$ is halting on s.

(17) Let s be a state of SCMPDS, I be a Program-block, a, c be Int positions, and i be an integer. If $s(\mathrm{DataLoc}(s(a), i)) = 0$, then $\mathrm{IExec}(\mathrm{while} <> 0(a, i, I), s) = s {+}\!\cdot \mathrm{Start\text{-}At}(\mathrm{inspos}\, \operatorname{card} I + 3)$.

(18) Let s be a state of SCMPDS, I be a Program-block, a be an Int position, and i be an integer. If $s(\mathrm{DataLoc}(s(a), i)) = 0$, then $\mathbf{IC}_{\mathrm{IExec}(\mathrm{while}<>0(a,i,I),s)} = \mathrm{inspos}\, \operatorname{card} I + 3$.

(19) Let s be a state of SCMPDS, I be a Program-block, a, b be Int positions, and i be an integer. If $s(\mathrm{DataLoc}(s(a), i)) = 0$, then $(\mathrm{IExec}(\mathrm{while} <> 0(a, i, I), s))(b) = s(b)$.

Let I be a shiftable Program-block, let a be an Int position, and let i be an integer. Observe that $\mathrm{while} <> 0(a, i, I)$ is shiftable.

Let I be a No-StopCode Program-block, let a be an Int position, and let i be an integer. Note that $\mathrm{while} <> 0(a, i, I)$ is No-StopCode.

8. Computing Directly the Result of "while<>0" Program by Loop-Invariant

Now we present three schemes. The scheme *WhileNHalt* deals with a unary functor $\mathcal{F}$ yielding a natural number, a state $\mathcal{A}$ of SCMPDS, a No-StopCode shiftable Program-block $\mathcal{B}$, an Int position $\mathcal{C}$, an integer $\mathcal{D}$, and a unary predicate $\mathcal{P}$, and states that:

$\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{A})$ or $\mathcal{P}[\mathcal{A}]$ but while $<> 0(\mathcal{C}, \mathcal{D}, \mathcal{B})$ is closed on $\mathcal{A}$ but while $<> 0(\mathcal{C}, \mathcal{D}, \mathcal{B})$ is halting on $\mathcal{A}$

provided the following conditions are satisfied:

- $\operatorname{card} \mathcal{B} > 0$,
- For every state t of SCMPDS such that $\mathcal{P}[\mathrm{Dstate}\, t]$ and $\mathcal{F}(\mathrm{Dstate}\, t) = 0$ holds $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) = 0$,
- $\mathcal{P}[\mathrm{Dstate}\, \mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\mathrm{Dstate}\, t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \neq 0$. Then $(\mathrm{IExec}(\mathcal{B}, t))(\mathcal{C}) = t(\mathcal{C})$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{F}(\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)) < \mathcal{F}(\mathrm{Dstate}\, t)$ and $\mathcal{P}[\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)]$.

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

$\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{A})$ or $\mathcal{P}[\mathcal{A}]$ but $\mathrm{IExec}(\text{while} <> 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathcal{A}) = \mathrm{IExec}(\text{while} <> 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathrm{IExec}(\mathcal{B}, \mathcal{A}))$

provided the parameters meet the following conditions:

- $\operatorname{card} \mathcal{B} > 0$,
- $\mathcal{A}(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \neq 0$,
- For every state t of SCMPDS such that $\mathcal{P}[\mathrm{Dstate}\, t]$ and $\mathcal{F}(\mathrm{Dstate}\, t) = 0$ holds $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) = 0$,
- $\mathcal{P}[\mathrm{Dstate}\, \mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\mathrm{Dstate}\, t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \neq 0$. Then $(\mathrm{IExec}(\mathcal{B}, t))(\mathcal{C}) = t(\mathcal{C})$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{F}(\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)) < \mathcal{F}(\mathrm{Dstate}\, t)$ and $\mathcal{P}[\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)]$.

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

$\mathcal{F}(\mathcal{A}) = \mathcal{F}(\mathcal{A})$ or $\mathcal{P}[\mathcal{A}]$ but $\mathcal{F}(\mathrm{Dstate}\, \mathrm{IExec}(\text{while} <> 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathcal{A})) = 0$ but $\mathcal{P}[\mathrm{Dstate}\, \mathrm{IExec}(\text{while} <> 0(\mathcal{C}, \mathcal{D}, \mathcal{B}), \mathcal{A})]$

provided the parameters satisfy the following conditions:

- $\operatorname{card} \mathcal{B} > 0$,
- For every state t of SCMPDS such that $\mathcal{P}[\mathrm{Dstate}\, t]$ holds $\mathcal{F}(\mathrm{Dstate}\, t) = 0$ iff $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) = 0$,
- $\mathcal{P}[\mathrm{Dstate}\, \mathcal{A}]$, and
- Let t be a state of SCMPDS. Suppose $\mathcal{P}[\mathrm{Dstate}\, t]$ and $t(\mathcal{C}) = \mathcal{A}(\mathcal{C})$ and $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \neq 0$. Then $(\mathrm{IExec}(\mathcal{B}, t))(\mathcal{C}) = t(\mathcal{C})$ and $\mathcal{B}$ is closed on t and $\mathcal{B}$ is halting on t and $\mathcal{F}(\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)) < \mathcal{F}(\mathrm{Dstate}\, t)$ and $\mathcal{P}[\mathrm{Dstate}\, \mathrm{IExec}(\mathcal{B}, t)]$.

We now state the proposition

(20) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, b, c be Int positions, and i, d be integers. Suppose that

(i) $\operatorname{card} I > 0$,

(ii) $s(a) = d$,

(iii) $s(b) > 0$,

(iv) $s(c) > 0$,

(v) $s(\mathrm{DataLoc}(d, i)) = s(b) - s(c)$, and

(vi) for every state t of SCMPDS such that $t(b) > 0$ and $t(c) > 0$ and $t(a) = d$ and $t(\mathrm{DataLoc}(d, i)) = t(b) - t(c)$ and $t(b) \neq t(c)$ holds $(\mathrm{IExec}(I, t))(a) = d$ and I is closed on t and halting on t and if $t(b) > t(c)$, then $(\mathrm{IExec}(I, t))(b) = t(b) - t(c)$ and $(\mathrm{IExec}(I, t))(c) = t(c)$ and if $t(b) \leqslant t(c)$, then $(\mathrm{IExec}(I, t))(c) = t(c) - t(b)$ and $(\mathrm{IExec}(I, t))(b) = t(b)$ and $(\mathrm{IExec}(I, t))(\mathrm{DataLoc}(d, i)) = (\mathrm{IExec}(I, t))(b) - (\mathrm{IExec}(I, t))(c)$.

Then while $<> 0(a, i, I)$ is closed on s and while $<> 0(a, i, I)$ is halting on s and if $s(\mathrm{DataLoc}(s(a), i)) \neq 0$, then $\mathrm{IExec}(\text{while} <> 0(a, i, I), s) = \mathrm{IExec}(\text{while} <> 0(a, i, I), \mathrm{IExec}(I, s))$.

9. An Example: Computing Greatest Common Divisor (Euclide Algorithm) by Loop-Invariant

The Program-block GCD-Algorithm is defined by:

(Def. 4) GCD-Algorithm $=$ (GBP :=0); ((GBP, 3) := (GBP, 1)); SubFrom(GBP, 3, GBP, 2); while $<> 0$(GBP, 3, (**if** GBP > 3 **then** Load(SubFrom(GBP, 1, GBP, 2)) **else** Load(SubFrom(GBP, 2, GBP, 1))); ((GBP, 3) := (GBP, 1)); SubFrom(GBP, 3, GBP, 2)).

Next we state the proposition

(21) Let s be a state of SCMPDS, I be a No-StopCode shiftable Program-block, a, b, c be Int positions, and i, d be integers. Suppose that

(i) $\operatorname{card} I > 0$,

(ii) $s(a) = d$,

(iii) $s(b) > 0$,

(iv) $s(c) > 0$,

(v) $s(\mathrm{DataLoc}(d, i)) = s(b) - s(c)$, and

(vi) for every state t of SCMPDS such that $t(b) > 0$ and $t(c) > 0$ and $t(a) = d$ and $t(\mathrm{DataLoc}(d, i)) = t(b) - t(c)$ and $t(b) \neq t(c)$ holds $(\mathrm{IExec}(I, t))(a) = d$ and I is closed on t and halting on t and if $t(b) > t(c)$, then $(\mathrm{IExec}(I, t))(b) = t(b) - t(c)$ and $(\mathrm{IExec}(I, t))(c) = t(c)$ and if $t(b) \leqslant t(c)$, then $(\mathrm{IExec}(I, t))(c) = t(c) - t(b)$ and $(\mathrm{IExec}(I, t))(b) = t(b)$ and $(\mathrm{IExec}(I, t))(\mathrm{DataLoc}(d, i)) = (\mathrm{IExec}(I, t))(b) - (\mathrm{IExec}(I, t))(c)$.

Then (IExec(while $<>$ $0(a, i, I), s))(b) = s(b) \gcd s(c)$ and (IExec(while $<>$ $0(a, i, I), s))(c) = s(b) \gcd s(c)$.

We now state the proposition

(22) card GCD-Algorithm $= 12$.

The following proposition is true

(23) Let s be a state of SCMPDS and x, y be integers. Suppose $s(\mathrm{intpos}\,1) = x$ and $s(\mathrm{intpos}\,2) = y$ and $x > 0$ and $y > 0$. Then (IExec(GCD-Algorithm, s))(intpos 1) $= x \gcd y$ and (IExec(GCD-Algorithm, s))(intpos 2) $= x \gcd y$ and GCD-Algorithm is closed on s and GCD-Algorithm is halting on s.

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. Two programs for **scm**. Part I - preliminaries. *Formalized Mathematics*, 4(**1**):69–72, 1993.
[6] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[7] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[8] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Formalized Mathematics*, 8(**1**):193–199, 1999.
[9] Jing-Chao Chen. Computation of two consecutive program blocks for SCMPDS. *Formalized Mathematics*, 8(**1**):211–217, 1999.
[10] Jing-Chao Chen. The construction and computation of conditional statements for SCMPDS. *Formalized Mathematics*, 8(**1**):219–234, 1999.
[11] Jing-Chao Chen. The construction and shiftability of program blocks for SCMPDS. *Formalized Mathematics*, 8(**1**):201–210, 1999.
[12] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Formalized Mathematics*, 8(**1**):183–191, 1999.
[13] Jing-Chao Chen. The construction and computation of while-loop programs for SCMPDS. *Formalized Mathematics*, 9(**2**):397–405, 2001.
[14] Jing-Chao Chen. Insert sort on SCMPDS. *Formalized Mathematics*, 9(**2**):407–412, 2001.
[15] Jing-Chao Chen. Recursive Euclide algorithm. *Formalized Mathematics*, 9(**1**):1–4, 2001.
[16] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[17] Andrzej Kondracki. The Chinese Remainder Theorem. *Formalized Mathematics*, 6(**4**):573–577, 1997.
[18] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[19] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[20] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[21] Andrzej Trybulec and Czesław Byliński. Some properties of real numbers. *Formalized Mathematics*, 1(**3**):445–449, 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] Wojciech A. Trybulec. Groups. *Formalized Mathematics*, 1(**5**):821–827, 1990.
[25] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[26] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[27] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received June 14, 2000