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

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

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

## 1. Preliminaries

For simplicity, we adopt the following convention: $m$, $n$ denote natural numbers, $i$, $j$ denote instructions of SCMPDS, $I$ denotes a Program-block, and $a$ denotes an Int position.

The following propositions are true:

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

(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 $a \geq 0$ and $b \geq 0$ holds $a \gcd b = a \gcd b - a$.

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

(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 $f(s) = \max(|s(a)|,|s(b)|)$.

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

(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 $s(a) \geq 0$, 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:

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

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 $t(\text{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \geq 0$,
- $\mathcal{P}[\text{Dstate}\,\mathcal{A}]$, 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 $\text{sum}(n, p_0)$ yielding a Program-block is defined by:

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

The following proposition is true

(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 $\text{card}\,I > 0$ and $f$ is FinSequence on $s$, $p_0$ and $\text{len}\,f = n$ and $s(b) = 0$ and $s(a) = 0$ and $s(\text{intpos}\,i) = -n$ 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 $\text{len}\,g = t(\text{intpos}\,i) + n$ and $t(b) = \sum g$ and $t(c) = p_0 + 1 + \text{len}\,g$ and $t(a) = 0$ and $t(\text{intpos}\,i) < 0$ and for every natural number $i$ such that $i > p_0$ holds $t(\text{intpos}\,i) = s(\text{intpos}\,i)$ holds $(\text{IExec}(I, t))(a) = 0$ and $I$ is closed on $t$ and halting on $t$ and $(\text{IExec}(I, t))(\text{intpos}\,i) = t(\text{intpos}\,i) + 1$ and there exists a finite sequence $g$ of elements of $\mathbb{Z}$ such that $g$ is FinSequence on $s$, $p_0$ and $\text{len}\,g = t(\text{intpos}\,i) + n + 1$ and $(\text{IExec}(I, t))(c) = p_0 + 1 + \text{len}\,g$ and $(\text{IExec}(I, t))(b) = \sum g$ and for every natural number $i$ such that $i > p_0$ holds $(\text{IExec}(I, t))(\text{intpos}\,i) = s(\text{intpos}\,i)$. Then $(\text{IExec}(\text{while} < 0(a, i, I), s))(b) = \sum f$ and $\text{while} < 0(a, i, I)$ is closed on $s$ and $\text{while} < 0(a, i, I)$ is halting on $s$.

The following proposition is true

(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 $p_0 \geq 3$ and $f$ is FinSequence on $s$, $p_0$ and $\text{len}\,f = n$. Then $(\text{IExec}(\text{sum}(n, p_0), s))(\text{intpos}\,1) = \sum f$ and $\text{sum}(n, p_0)$ 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:

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

provided the following conditions are met:

- $\text{card}\,\mathcal{B} > 0$,

- For every state $t$ of SCMPDS such that $\mathcal{P}[\text{Dstate}\,t]$ holds $\mathcal{F}(\text{Dstate}\,t) = 0$ iff $t(\text{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) \leq 0$,
- $\mathcal{P}[\text{Dstate}\,\mathcal{A}]$, 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 $\mathcal{P}[\text{Dstate}\,\text{IExec}(\mathcal{B},t)]$.

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

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

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

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) $\text{card}\,I > 0$,

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

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

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

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

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

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

The following proposition is true

(11) For every state $s$ of SCMPDS and for every natural number $n$ holds $(\text{IExec}(\text{Fib-macro}\,n,s))(\text{intpos}\,1) = \text{Fib}(n)$ and $(\text{IExec}(\text{Fib-macro}\,n,s))(\text{intpos}\,2) = \text{Fib}(n+1)$ 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 $<> 0(a,i,I)$ yielding a Program-block is defined by:

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

## 7. The Basic Property of “while<>0” Program

Next we state several propositions:

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

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

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

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

(16) Let $s$ be a state of SCMPDS, $I$ be a Program-block, $a$ be an Int position, and $i$ be an integer. If $s(\text{DataLoc}(s(a),i)) = 0$, then while $<> 0(a,i,I)$ is closed on $s$ and 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 $s(\text{DataLoc}(s(a),i)) = 0$, then $\text{IExec}(\text{while} <> 0(a,i,I),s) = s +\cdot \text{Start-At}(\text{inspos}\, \text{card}\, I + 3)$.

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

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

Let $I$ be a shiftable Program-block, let $a$ be an Int position, and let $i$ be an integer. Note that 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. One can check that 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:

while $<> 0(\mathcal{C},\mathcal{D},\mathcal{B})$ is closed on $\mathcal{A}$ and while $<> 0(\mathcal{C},\mathcal{D},\mathcal{B})$ is halting on $\mathcal{A}$

provided the parameters have the following properties:

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

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:

$\text{IExec}(\text{while} <> 0(\mathcal{C},\mathcal{D},\mathcal{B}),\mathcal{A}) = \text{IExec}(\text{while} <> 0(\mathcal{C},\mathcal{D},\mathcal{B}),\text{IExec}(\mathcal{B},\mathcal{A}))$

provided the parameters satisfy the following conditions:

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

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:

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

provided the following conditions are satisfied:

- $\text{card}\, \mathcal{B} > 0$,
- For every state $t$ of SCMPDS such that $\mathcal{P}[\text{Dstate}\, t]$ holds $\mathcal{F}(\text{Dstate}\, t) = 0$ iff $t(\text{DataLoc}(\mathcal{A}(\mathcal{C}),\mathcal{D})) = 0$,

- $\mathcal{P}$[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 $t(\mathrm{DataLoc}(\mathcal{A}(\mathcal{C}), \mathcal{D})) \neq 0$. 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 $\mathcal{P}$[Dstate IExec$(\mathcal{B},t)$].

One can prove the following 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) $\mathrm{card}\, I > 0$,

(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 $t(b) \leq t(c)$, 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)).

One can prove the following 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) $\mathrm{card}\, I > 0$,

(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 $t(b) \leq t(c)$, 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 $(\mathrm{IExec}(\text{while} <> 0(a,i,I),s))(b) = s(b) \gcd s(c)$ and $(\mathrm{IExec}(\text{while} <> 0(a,i,I),s))(c) = s(b) \gcd s(c)$.

Next we state the proposition

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

One can prove the following proposition

(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 $(\mathrm{IExec}(\text{GCD-Algorithm}, s))(\mathrm{intpos}\,1) = x \,\mathrm{gcd}\, y$ and $(\mathrm{IExec}(\text{GCD-Algorithm}, s))(\mathrm{intpos}\,2) = x \,\mathrm{gcd}\, y$ and GCD-Algorithm is closed on $s$ and GCD-Algorithm is halting on $s$.

## 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. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/card_3.html.

[4] Grzegorz Bancerek. Joining of decorated trees. *Journal of Formalized Mathematics*, 5, 1993. http://mizar.org/JFM/Vol5/trees_4.html.

[5] 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.

[6] Grzegorz Bancerek and Piotr Rudnicki. Two programs for **scm**. Part I - preliminaries. *Journal of Formalized Mathematics*, 5, 1993. http://mizar.org/JFM/Vol5/pre_ff.html.

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

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

[9] Czesław Byliński. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funct_2.html.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] Jing-Chao Chen. Insert sort on SCMPDS. *Journal of Formalized Mathematics*, 12, 2000. http://mizar.org/JFM/Vol12/scpisort.html.

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

[20] Rafał Kwiatek and Grzegorz Zwara. The divisibility of integers and integer relatively primes. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/int_2.html.

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

[22] 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.

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

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

[25] Andrzej Trybulec and Czesław Byliński. Some properties of real numbers operations: min, max, square, and square root. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/square_1.html.

[26] 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.

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

[28] Wojciech A. Trybulec. Groups. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/group_1.html.

[29] Zinaida Trybulec. Properties of subsets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/subset_1.html.

[30] 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*