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

Recursive Euclide Algorithm [1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. The earlier SCM computer did not contain recursive function, so Trybulec and Nakamura proved the correctness of the Euclid's algorithm only by way of an iterative program. However, the recursive method is a very important programming method, furthermore, for some algorithms, for example Quicksort, only by employing a recursive method (note push-down stack is essentially also a recursive method) can they be implemented. The main goal of the article is to test the recursive function of the SCMPDS computer by proving the correctness of the Euclid's algorithm by way of a recursive program. In this article, we observed that the memory required by the recursive Euclide algorithm is variable but it is still autonomic. Although the algorithm here is more complicated than the non-recursive algorithm, its focus is that the SCMPDS computer will be able to implement many algorithms like Quicksort which the SCM computer cannot do.

MML Identifier: `SCMP_GCD`.

WWW: `http://mizar.org/JFM/Vol11/scmp_gcd.html`

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

1. PRELIMINARIES

For simplicity, we use the following convention: m, n denote natural numbers, i, j denote instructions of SCMPDS, s denotes a state of SCMPDS, and I, J denote Program-blocks.

The following three propositions are true:

(1) If $m > 0$, then $\gcd(n,m) = \gcd(m, n \bmod m)$.

(2) For all integers i, j such that $i \geq 0$ and $j > 0$ holds $i \gcd j = j \gcd i \bmod j$.

(3) For every natural number m and for every integer j such that $\mathrm{inspos}\, m = j$ holds $\mathrm{inspos}\, m + 2 = 2 \cdot (|j| \div 2) + 4$.

Let k be a natural number. The functor $\mathrm{intpos}\, k$ yields an Int position and is defined by:

(Def. 1) $\mathrm{intpos}\, k = \mathbf{d}_k$.

We now state three propositions:

(4) For all natural numbers n_1, n_2 such that $n_1 \neq n_2$ holds $\mathrm{intpos}\, n_1 \neq \mathrm{intpos}\, n_2$.

(5) For all natural numbers n_1, n_2 holds $\mathrm{DataLoc}(n_1, n_2) = \mathrm{intpos}\, n_1 + n_2$.

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

(6) For every state s of SCMPDS and for all natural numbers m_1, m_2 such that $\mathbf{IC}_s = \mathrm{inspos}\, m_1 + m_2$ holds $\mathrm{ICplusConst}(s, -m_2) = \mathrm{inspos}\, m_1$.

The Int position GBP is defined as follows:

(Def. 2) $\mathrm{GBP} = \mathrm{intpos}\, 0$.

The Int position SBP is defined by:

(Def. 3) $\mathrm{SBP} = \mathrm{intpos}\, 1$.

We now state several propositions:

(7) $\mathrm{GBP} \neq \mathrm{SBP}$.

(8) $\mathrm{card}(I;\, i) = \mathrm{card}\, I + 1$.

(9) $\mathrm{card}(i;\, j) = 2$.

(10) $(I;\, i)(\mathrm{inspos}\,\mathrm{card}\, I) = i$ and $\mathrm{inspos}\,\mathrm{card}\, I \in \mathrm{dom}(I;\, i)$.

(11) $(I;\, i;\, J)(\mathrm{inspos}\,\mathrm{card}\, I) = i$.

2. The Construction of Recursive Euclide Algorithm

The Program-block GCD-Algorithm is defined by:

(Def. 4) GCD-Algorithm $=$ (GBP:=0); (SBP:=7); saveIC(SBP, RetIC); goto 2; $\mathbf{halt}_{\mathrm{SCMPDS}}$; ((SBP, 3) <= 0_goto 9); ((SBP, 6) := (SBP, 3)); Divide(SBP, 2, SBP, 3); ((SBP, 7) := (SBP, 3)); ((SBP, 4 + RetSP) := (GBP, 1)); AddTo(GBP, 1, 4); saveIC(SBP, RetIC); goto (-7); ((SBP, 2) := (SBP, 6)); return SBP.

3. The Computation of Recursive Euclide Algorithm

One can prove the following propositions:

(12) card GCD-Algorithm $= 15$.

(13) $n < 15$ iff $\mathrm{inspos}\, n \in \mathrm{dom}$ GCD-Algorithm.

(14) (GCD-Algorithm)(inspos 0) $=$ GBP:=0 and (GCD-Algorithm)(inspos 1) $=$ SBP:=7 and (GCD-Algorithm)(inspos 2) $=$ saveIC(SBP, RetIC) and (GCD-Algorithm)(inspos 3) $=$ goto 2 and (GCD-Algorithm)(inspos 4) $= \mathbf{halt}_{\mathrm{SCMPDS}}$ and (GCD-Algorithm)(inspos 5) $=$ (SBP, 3) <= 0_goto 9 and (GCD-Algorithm)(inspos 6) $=$ (SBP, 6) := (SBP, 3) and (GCD-Algorithm)(inspos 7) $=$ Divide(SBP, 2, SBP, 3) and (GCD-Algorithm)(inspos 8) $=$ (SBP, 7) := (SBP, 3) and (GCD-Algorithm)(inspos 9) $=$ (SBP, 4 + RetSP) := (GBP, 1) and (GCD-Algorithm)(inspos 10) $=$ AddTo(GBP, 1, 4) and (GCD-Algorithm)(inspos 11) $=$ saveIC(SBP, RetIC) and (GCD-Algorithm)(inspos 12) $=$ goto (-7) and (GCD-Algorithm)(inspos 13) $=$ (SBP, 2) := (SBP, 6) and (GCD-Algorithm)(inspos 14) $=$ return SBP.

(15) Let s be a state of SCMPDS. Suppose Initialized(GCD-Algorithm) $\subseteq s$. Then $\mathbf{IC}_{(\mathrm{Computation}(s))(4)} = \mathrm{inspos}\, 5$ and $(\mathrm{Computation}(s))(4)(\mathrm{GBP}) = 0$ and $(\mathrm{Computation}(s))(4)(\mathrm{SBP}) = 7$ and $(\mathrm{Computation}(s))(4)(\mathrm{intpos}\, 7 + \mathrm{RetIC}) = \mathrm{inspos}\, 2$ and $(\mathrm{Computation}(s))(4)(\mathrm{intpos}\, 9) = s(\mathrm{intpos}\, 9)$ and $(\mathrm{Computation}(s))(4)(\mathrm{intpos}\, 10) = s(\mathrm{intpos}\, 10)$.

(16) Let s be a state of SCMPDS. Suppose GCD-Algorithm $\subseteq s$ and $\mathbf{IC}_s = \mathrm{inspos}\, 5$ and $s(\mathrm{SBP}) > 0$ and $s(\mathrm{GBP}) = 0$ and $s(\mathrm{DataLoc}(s(\mathrm{SBP}), 3)) \geq 0$ and $s(\mathrm{DataLoc}(s(\mathrm{SBP}), 2)) \geq s(\mathrm{DataLoc}(s(\mathrm{SBP}), 3))$. Then there exists n such that

(i) $\mathrm{CurInstr}((\mathrm{Computation}(s))(n)) = \mathrm{return}\, \mathrm{SBP}$,

(ii) $s(\mathrm{SBP}) = (\mathrm{Computation}(s))(n)(\mathrm{SBP})$,

(iii) $(\mathrm{Computation}(s))(n)(\mathrm{DataLoc}(s(\mathrm{SBP}),2)) = s(\mathrm{DataLoc}(s(\mathrm{SBP}),2)) \gcd s(\mathrm{DataLoc}(s(\mathrm{SBP}),3))$, and

(iv) for every natural number j such that $1 < j$ and $j \leq s(\mathrm{SBP})+1$ holds $s(\mathrm{intpos}\, j) = (\mathrm{Computation}(s))(n)(\mathrm{intpos}\, j)$.

(17) Let s be a state of SCMPDS. Suppose GCD-Algorithm $\subseteq s$ and $\mathbf{IC}_s = \mathrm{inspos}\, 5$ and $s(\mathrm{SBP}) > 0$ and $s(\mathrm{GBP}) = 0$ and $s(\mathrm{DataLoc}(s(\mathrm{SBP}),3)) \geq 0$ and $s(\mathrm{DataLoc}(s(\mathrm{SBP}),2)) \geq 0$. Then there exists n such that

(i) $\mathrm{CurInstr}((\mathrm{Computation}(s))(n)) = \mathrm{return}\, \mathrm{SBP}$,

(ii) $s(\mathrm{SBP}) = (\mathrm{Computation}(s))(n)(\mathrm{SBP})$,

(iii) $(\mathrm{Computation}(s))(n)(\mathrm{DataLoc}(s(\mathrm{SBP}),2)) = s(\mathrm{DataLoc}(s(\mathrm{SBP}),2)) \gcd s(\mathrm{DataLoc}(s(\mathrm{SBP}),3))$, and

(iv) for every natural number j such that $1 < j$ and $j \leq s(\mathrm{SBP})+1$ holds $s(\mathrm{intpos}\, j) = (\mathrm{Computation}(s))(n)(\mathrm{intpos}\, j)$.

4. The Correctness of Recursive Euclide Algorithm

One can prove the following proposition

(18) Let s be a state of SCMPDS. Suppose Initialized(GCD-Algorithm) $\subseteq s$. Let x, y be integers. If $s(\mathrm{intpos}\, 9) = x$ and $s(\mathrm{intpos}\, 10) = y$ and $x \geq 0$ and $y \geq 0$, then $(\mathrm{Result}(s))(\mathrm{intpos}\, 9) = x \gcd y$.

5. The Autonomy of Recursive Euclide Algorithm

One can prove the following proposition

(19) Let p be a finite partial state of SCMPDS and x, y be integers. If $y \geq 0$ and $x \geq y$ and $p = [\mathrm{intpos}\, 9 \longmapsto x, \mathrm{intpos}\, 10 \longmapsto y]$, then Initialized(GCD-Algorithm)$+\cdot p$ is autonomic.

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

[4] 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`.

[5] 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`.

[6] 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`.

[7] 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`.

[8] Jing-Chao Chen. A small computer model with push-down stack. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_1.html`.

[9] 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`.

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

[11] 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`.

[12] 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`.

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

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

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

Received June 15, 1999

Published January 2, 2004