FORMALIZED MATHEMATICS
Volume 9, Number 1, 2001
University 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.

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

1. Preliminaries

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

One can prove the following three propositions:

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

(2) For all integers i, j such that $i \geqslant 0$ and $j > 0$ holds $i \gcd j = j \gcd i \mathrm{mod} j$.

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

ISSN 1426–2630

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

Let k be a natural number. The functor intpos k yields a Int position and is defined as follows:

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

Next we state three propositions:

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

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

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

The Int position GBP is defined by:

(Def. 2) GBP = intpos 0.

The Int position SBP is defined as follows:

(Def. 3) SBP = intpos 1.

The following propositions are true:

(7) GBP $\neq$ SBP .

(8) card$(I;i) =$ card $I + 1$.

(9) card$(i;j) = 2$.

(10) $(I;i)$(inspos card $I) = i$ and inspos card $I \in$ dom$(I;i)$.

(11) $(I;i;J)$(inspos 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; $\mathbf{halt}_{\rm SCMPDS}$;((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 inspos $n \in$ 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) = $\mathbf{halt}_{\rm SCMPDS}$ 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 $\mathbf{IC}_{({\rm Computation}(s))(4)}$ = inspos 5 and (Computation(s))(4)(GBP) = 0 and (Computation(s))(4)(SBP) = 7 and (Computation(s))(4)(intpos 7+ RetIC) = inspos 2 and (Computation(s))(4)(intpos 9) = s(intpos 9) and (Computation(s))(4)(intpos 10) = s(intpos 10).

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

(i) CurInstr((Computation(s))(n)) = return SBP,

(ii) s(SBP) = (Computation(s))(n)(SBP),

(iii) (Computation(s))(n)(DataLoc(s(SBP), 2)) = s(DataLoc(s(SBP), 2)) gcd s(DataLoc(s(SBP), 3)), and

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

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

(i) CurInstr((Computation(s))(n)) = return SBP,

(ii) s(SBP) = (Computation(s))(n)(SBP),

(iii) (Computation(s))(n)(DataLoc(s(SBP), 2)) = s(DataLoc(s(SBP), 2)) gcd s(DataLoc(s(SBP), 3)), and

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

4. The Correctness of Recursive Euclide Algorithm

The following proposition is true

(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 $x \geqslant 0$ and $y \geqslant 0$, then $(\mathrm{Result}(s))(\mathrm{intpos}\,9) = x \gcd y$.

5. The Autonomy of Recursive Euclide Algorithm

We now state the proposition

(19) Let p be a finite partial state of SCMPDS and x, y be integers. If $y \geqslant 0$ and $x \geqslant y$ 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. *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] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[5] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.
[6] Jing-Chao Chen. Computation and program shift in the SCMPDS computer. *Formalized Mathematics*, 8(**1**):193–199, 1999.
[7] Jing-Chao Chen. The construction and shiftability of program blocks for SCMPDS. *Formalized Mathematics*, 8(**1**):201–210, 1999.
[8] Jing-Chao Chen. The SCMPDS computer and the basic semantics of its instructions. *Formalized Mathematics*, 8(**1**):183–191, 1999.
[9] Jing-Chao Chen. A small computer model with push-down stack. *Formalized Mathematics*, 8(**1**):175–182, 1999.
[10] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[11] Rafał Kwiatek and Grzegorz Zwara. The divisibility of integers and integer relative primes. *Formalized Mathematics*, 1(**5**):829–832, 1990.
[12] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[13] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[14] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[15] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[16] Wojciech A. Trybulec. Groups. *Formalized Mathematics*, 1(**5**):821–827, 1990.
[17] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[18] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received June 15, 1999