FORMALIZED MATHEMATICS
Volume 4, Number 1, 1993
Université Catholique de Louvain

Some Remarks on the Simple Concrete Model of Computer

Andrzej Trybulec
Warsaw University
Białystok

Yatsuka Nakamura
Shinshu University
Nagano

Summary. We prove some results on **SCM** needed for the proof of the correctness of Euclid's algorithm. We introduce the following concepts:

- starting finite partial state (Start-At(l)), then assigns to the instruction counter an instruction location (and consists only of this assignment),
- programmed finite partial state, that consists of the instructions (to be more precise, a finite partial state with the domain consisting of instruction locations).

We define for a total state s what it means that s starts at l (the value of the instruction counter in the state s is l) and s halts at l (the halt instruction is assigned to l in the state s). Similar notions are defined for finite partial states.

MML Identifier: AMI_3.

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

1. Preliminaries

One can prove the following proposition

(1) For all integers m, j holds $m \cdot j \equiv +0 (\mathrm{mod}\, m)$.

In the sequel i, j, k will denote natural numbers.

The scheme *INDI* concerns natural numbers $\mathcal{A}$, $\mathcal{B}$ and a unary predicate $\mathcal{P}$, and states that:

ISSN 0777-4028

$\mathcal{P}[\mathcal{B}]$

provided the following requirements are met:

- $\mathcal{P}[0]$,
- $\mathcal{A} > 0$,
- For all i, j such that $\mathcal{P}[\mathcal{A} \cdot i]$ and $j \neq 0$ and $j \leq \mathcal{A}$ holds $\mathcal{P}[\mathcal{A} \cdot i + j]$.

In the sequel x will be arbitrary.

Next we state a number of propositions:

(2) Let X, Y be non empty set and let f, g be partial functions from X to Y. Suppose that for every element x of X and for every element y of Y holds $\langle x, y\rangle \in f$ iff $\langle x, y\rangle \in g$. Then $f = g$.

(3) For all functions f, g and for all sets A, B such that $f \restriction A = g \restriction A$ and $f \restriction B = g \restriction B$ holds $f \restriction (A \cup B) = g \restriction (A \cup B)$.

(4) For every set X and for all functions f, g such that $\operatorname{dom} g \subseteq X$ and $g \subseteq f$ holds $g \subseteq f \restriction X$.

(5) For every function f and for arbitrary x such that $x \in \operatorname{dom} f$ holds $f \restriction \{x\} = \{\langle x, f(x)\rangle\}$.

(6) For every function f and for every set X such that $X \cap \operatorname{dom} f = \emptyset$ holds $f \restriction X = \emptyset$.

(7) For all functions f, g and for arbitrary x such that $\operatorname{dom} f = \operatorname{dom} g$ and $f(x) = g(x)$ holds $f \restriction \{x\} = g \restriction \{x\}$.

(8) For all functions f, g and for arbitrary x, y such that $\operatorname{dom} f = \operatorname{dom} g$ and $f(x) = g(x)$ and $f(y) = g(y)$ holds $f \restriction \{x, y\} = g \restriction \{x, y\}$.

(9) Let f, g be functions and let x, y, z be arbitrary. If $\operatorname{dom} f = \operatorname{dom} g$ and $f(x) = g(x)$ and $f(y) = g(y)$ and $f(z) = g(z)$, then $f \restriction \{x, y, z\} = g \restriction \{x, y, z\}$.

(10) For arbitrary a, b and for every function f such that $a \in \operatorname{dom} f$ and $f(a) = b$ holds $a \mapsto b \subseteq f$.

(11) For arbitrary a, b, c, d such that $a \neq c$ holds $[a \longmapsto b, c \longmapsto d] = \{\langle a, b\rangle, \langle c, d\rangle\}$.

(12) For arbitrary a, b, c, d and for every function f such that $a \in \operatorname{dom} f$ and $c \in \operatorname{dom} f$ and $f(a) = b$ and $f(c) = d$ holds $[a \longmapsto b, c \longmapsto d] \subseteq f$.

(13) For all functions f, g, h holds $(f +\cdot g) +\cdot h = f +\cdot (g +\cdot h)$.

2. Computations

In the sequel N denotes a non empty set with non empty elements.

Next we state the proposition

(14) For every AMI S over N and for every finite partial state p of S holds $p \in \operatorname{FinPartSt}(S)$.

Let us consider N and let S be an AMI over N. Then $\operatorname{FinPartSt}(S)$ is a non empty subset of $\prod^{\cdot}$(the object kind of S).

Next we state two propositions:

(15) For every AMI S over N holds every element of FinPartSt(S) is a finite partial state of S.

(16) Let S be an AMI over N and let F_1, F_2 be partial functions from FinPartSt(S) to FinPartSt(S). Suppose that for all finite partial states p, q of S holds $\langle p, q \rangle \in F_1$ iff $\langle p, q \rangle \in F_2$. Then $F_1 = F_2$.

The scheme *EqFPSFunc* concerns a non empty set $\mathcal{A}$ with non empty elements, an AMI $\mathcal{B}$ over $\mathcal{A}$, partial functions $\mathcal{C}$, $\mathcal{D}$ from FinPartSt($\mathcal{B}$) to FinPartSt($\mathcal{B}$), and a binary predicate $\mathcal{P}$, and states that:

$\mathcal{C} = \mathcal{D}$

provided the parameters meet the following conditions:

- For all finite partial states p, q of $\mathcal{B}$ holds $\langle p, q \rangle \in \mathcal{C}$ iff $\mathcal{P}[p, q]$,
- For all finite partial states p, q of $\mathcal{B}$ holds $\langle p, q \rangle \in \mathcal{D}$ iff $\mathcal{P}[p, q]$.

Let us consider N, let S be a von Neumann definite AMI over N, and let l be an instruction-location of S. The functor Start-At(l) yielding a finite partial state of S is defined by:

(Def.1) Start-At(l) = $\mathbf{IC}_S \longmapsto l$.

One can prove the following proposition

(17) For every von Neumann definite AMI S over N and for every instruction-location l of S holds dom Start-At(l) = $\{\mathbf{IC}_S\}$.

Let us consider N and let S be an AMI over N. A finite partial state of S is programmed if:

(Def.2) dom it $\subseteq$ the instruction locations of S.

We now state four propositions:

(18) Let S be a steady-programmed von Neumann definite AMI over N and let p_1, p_2 be programmed finite partial state of S. Then $p_1 +\cdot\, p_2$ is programmed.

(19) For every AMI S over N and for every state s of S holds dom s = the objects of S.

(20) For every AMI S over N and for every finite partial state p of S holds dom $p \subseteq$ the objects of S.

(21) Let S be a steady-programmed von Neumann definite AMI over N, and let p be a programmed finite partial state of S, and let s be a state of S. If $p \subseteq s$, then for every k holds $p \subseteq$ (Computation(s))(k).

Let us consider N, let S be a von Neumann AMI over N, let s be a state of S, and let l be an instruction-location of S. We say that s starts at l if and only if:

(Def.3) $\mathbf{IC}_s = l$.

We say that s halts at l if and only if:

(Def.4) $s(l) = \mathbf{halt}_S$.

The following proposition is true

(22) For every AMI S over N and for every finite partial state p of S there exists a state s of S such that $p \subseteq s$.

Let us consider N, let S be a definite von Neumann AMI over N, and let p be a finite partial state of S. Let us assume that $\mathbf{IC}_S \in \operatorname{dom} p$. The functor $\mathbf{IC}_p$ yielding an instruction-location of S is defined by:

(Def.5) $\mathbf{IC}_p = p(\mathbf{IC}_S)$.

Let us consider N, let S be a definite von Neumann AMI over N, let p be a finite partial state of S, and let l be an instruction-location of S. We say that p starts at l if and only if:

(Def.6) $\mathbf{IC}_S \in \operatorname{dom} p$ and $\mathbf{IC}_p = l$.

We say that p halts at l if and only if:

(Def.7) $l \in \operatorname{dom} p$ and $p(l) = \mathbf{halt}_S$.

One can prove the following propositions:

(23) Let S be a von Neumann definite steady-programmed AMI over N and let s be a state of S. Then s is halting if and only if there exists k such that s halts at $\mathbf{IC}_{(\text{Computation}(s))(k)}$.

(24) Let S be a von Neumann definite steady-programmed AMI over N, and let s be a state of S, and let p be a finite partial state of S, and let l be an instruction-location of S. If $p \subseteq s$ and p halts at l, then s halts at l.

(25) Let S be a halting steady-programmed von Neumann definite AMI over N, and let s be a state of S, and given k. If s is halting, then $\text{Result}(s) = (\text{Computation}(s))(k)$ iff s halts at $\mathbf{IC}_{(\text{Computation}(s))(k)}$.

(26) Let S be a steady-programmed von Neumann definite AMI over N, and let s be a state of S, and let p be a programmed finite partial state of S, and given k. Then $p \subseteq s$ if and only if $p \subseteq (\text{Computation}(s))(k)$.

(27) Let S be a halting steady-programmed von Neumann definite AMI over N, and let s be a state of S, and given k. If s halts at $\mathbf{IC}_{(\text{Computation}(s))(k)}$, then $\text{Result}(s) = (\text{Computation}(s))(k)$.

(28) Suppose $i \leq j$. Let S be a halting steady-programmed von Neumann definite AMI over N and let s be a state of S. If s halts at $\mathbf{IC}_{(\text{Computation}(s))(i)}$, then s halts at $\mathbf{IC}_{(\text{Computation}(s))(j)}$.

(29) Suppose $i \leq j$. Let S be a halting steady-programmed von Neumann definite AMI over N and let s be a state of S. If s halts at $\mathbf{IC}_{(\text{Computation}(s))(i)}$, then $(\text{Computation}(s))(j) = (\text{Computation}(s))(i)$.

(30) Let S be a steady-programmed von Neumann halting definite AMI over N and let s be a state of S. If there exists k such that s halts at $\mathbf{IC}_{(\text{Computation}(s))(k)}$, then for every i holds $\text{Result}(s) = \text{Result}((\text{Computation}(s))(i))$.

(31) Let S be a steady-programmed von Neumann definite AMI over N, and let s be a state of S, and let l be an instruction-location of S, and given k. Then s halts at l if and only if $(\text{Computation}(s))(k)$ halts at l.

(32) Let S be a definite von Neumann AMI over N, and let p be a finite partial state of S, and let l be an instruction-location of S. Suppose p starts at l. Let s be a state of S. If $p \subseteq s$, then s starts at l.

(33) For every von Neumann definite AMI S over N and for every instruction-location l of S holds Start-At$(l)(\mathbf{IC}_S) = l$.

Let us consider N, let S be a definite von Neumann AMI over N, let l be an instruction-location of S, and let I be an instruction of S. Then $l \dot{\longmapsto} I$ is a programmed finite partial state of S.

3. Instruction Locations and Data Locations

We now state the proposition

(34) **SCM** is realistic.

SCM is a steady-programmed halting realistic von Neumann data-oriented definite strict AMI over $\{\mathbb{Z}\}$.

Let us consider k. The functor $\mathbf{d}_k$ yields a data-location and is defined by:

(Def.8) $\mathbf{d}_k = 2 \cdot k + 1$.

The functor $\mathbf{i}_k$ yielding an instruction-location of **SCM** is defined by:

(Def.9) $\mathbf{i}_k = 2 \cdot k + 2$.

Next we state three propositions:

(35) For all i, j such that $i \neq j$ holds $\mathbf{d}_i \neq \mathbf{d}_j$.

(36) For all i, j such that $i \neq j$ holds $\mathbf{i}_i \neq \mathbf{i}_j$.

(37) $\mathrm{Next}(\mathbf{i}_k) = \mathbf{i}_{k+1}$.

Let s be a state of **SCM** and let a be a data-location. Then $s(a)$ is an integer.

Let us consider a, b. Then $a{:=}b$ is an instruction of **SCM**. Then AddTo(a, b) is an instruction of **SCM**. Then SubFrom(a, b) is an instruction of **SCM**. Then MultBy(a, b) is an instruction of **SCM**. Then Divide(a, b) is an instruction of **SCM**.

Let us consider l_1. Then goto l_1 is an instruction of **SCM**. Let us consider a. Then **if** $a = 0$ **goto** l_1 is an instruction of **SCM**. Then **if** $a > 0$ **goto** l_1 is an instruction of **SCM**.

Next we state the proposition

(38) For every data-location l holds ObjectKind$(l) = \mathbb{Z}$.

Let l_2 be a data-location and let a be an integer. Then $l_2 \dot{\longmapsto} a$ is a finite partial state of **SCM**.

Let l_2, l_3 be data-locations and let a, b be integers. Then $[l_2 \longmapsto a, l_3 \longmapsto b]$ is a finite partial state of **SCM**.

Next we state two propositions:

(39) For all i, j holds $\mathbf{d}_i \neq \mathbf{i}_j$.

(40) For every i holds $\mathbf{IC}_{\mathbf{SCM}} \neq \mathbf{d}_i$ and $\mathbf{IC}_{\mathbf{SCM}} \neq \mathbf{i}_i$.

References

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
[2] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.
[3] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[4] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.
[5] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[6] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[7] Czesław Byliński. Graphs of functions. *Formalized Mathematics*, 1(**1**):169–173, 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] Czesław Byliński. Partial functions. *Formalized Mathematics*, 1(**2**):357–367, 1990.
[10] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.
[11] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.
[12] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.
[13] Rafał Kwiatek and Grzegorz Zwara. The divisibility of integers and integer relative primes. *Formalized Mathematics*, 1(**5**):829–832, 1990.
[14] Michał Muzalewski. Rings and modules - part II. *Formalized Mathematics*, 2(**4**):579–585, 1991.
[15] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[16] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[17] Jan Popiołek. Some properties of functions modul and signum. *Formalized Mathematics*, 1(**2**):263–264, 1990.
[18] Dariusz Surowik. Cyclic groups and some of their properties - part I. *Formalized Mathematics*, 2(**5**):623–627, 1991.
[19] Andrzej Trybulec. Binary operations applied to functions. *Formalized Mathematics*, 1(**2**):329–334, 1990.
[20] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[21] Andrzej Trybulec. Function domains and Frænkel operator. *Formalized Mathematics*, 1(**3**):495–500, 1990.
[22] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[23] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[24] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received October 8, 1993