FORMALIZED MATHEMATICS
Volume 9, Number 4, 2001
University of Białystok

Input and Output of Instructions[1]

Artur Korniłowicz
University of Białystok

MML Identifier: AMI_7.

The terminology and notation used here are introduced in the following articles: [10], [5], [9], [6], [13], [1], [7], [4], [2], [11], [3], [12], and [8].

1. Preliminaries

In this paper N is a set with non empty elements.

One can prove the following propositions:

(1) For all sets x, y, z such that $x \neq y$ and $x \neq z$ holds $\{x, y, z\} \setminus \{x\} = \{y, z\}$.

(2) For every non empty non void AMI A over N and for every state s of A and for every object o of A holds $s(o) \in \mathrm{ObjectKind}(o)$.

(3) Let A be a realistic IC-Ins-separated definite non empty non void AMI over N, s be a state of A, f be an instruction-location of A, and w be an element of ObjectKind($\mathbf{IC}_A$). Then $(s +\cdot (\mathbf{IC}_A, w))(f) = s(f)$.

Let N be a set with non empty elements, let A be an IC-Ins-separated definite non empty non void AMI over N, let s be a state of A, let o be an object of A, and let a be an element of ObjectKind(o). Then $s +\cdot (o, a)$ is a state of A.

We now state several propositions:

(4) Let A be a steady-programmed IC-Ins-separated definite non empty non void AMI over N, s be a state of A, o be an object of A, f be an instruction-location of A, I be an instruction of A, and w be an element of ObjectKind(o). If $f \neq o$, then $(\mathrm{Exec}(I, s))(f) = (\mathrm{Exec}(I, s +\cdot (o, w)))(f)$.

[1]This work has been partially supported by TYPES grant IST-1999-29001.

ISSN 1426–2630

(5) Let A be an IC-Ins-separated definite non empty non void AMI over N, s be a state of A, o be an object of A, and w be an element of ObjectKind(o). If $o \neq \mathbf{IC}_A$, then $\mathbf{IC}_s = \mathbf{IC}_{s+\cdot(o,w)}$.

(6) Let A be a standard IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, s be a state of A, o be an object of A, and w be an element of ObjectKind(o). If I is sequential and $o \neq \mathbf{IC}_A$, then $\mathbf{IC}_{\mathrm{Exec}(I,s)} = \mathbf{IC}_{\mathrm{Exec}(I,s+\cdot(o,w))}$.

(7) Let A be a standard IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, s be a state of A, o be an object of A, and w be an element of ObjectKind(o). If I is sequential and $o \neq \mathbf{IC}_A$, then $\mathbf{IC}_{\mathrm{Exec}(I,s+\cdot(o,w))} = \mathbf{IC}_{\mathrm{Exec}(I,s)+\cdot(o,w)}$.

(8) Let A be a standard steady-programmed IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, s be a state of A, o be an object of A, w be an element of ObjectKind(o), and i be an instruction-location of A. Then $(\mathrm{Exec}(I, s +\cdot (o,w)))(i) = (\mathrm{Exec}(I,s) +\cdot (o,w))(i)$.

2. Input and Output of Instructions

Let N be a set and let A be an AMI over N. We say that A has non trivial instruction set if and only if:

(Def. 1) The instructions of A are non trivial.

Let N be a set and let A be a non empty AMI over N. We say that A has non trivial ObjectKinds if and only if:

(Def. 2) For every object o of A holds ObjectKind(o) is non trivial.

Let N be a set with non empty elements. One can verify that STC(N) has non trivial ObjectKinds.

Let N be a set with non empty elements. Observe that there exists a regular standard IC-Ins-separated definite non empty non void AMI over N which is halting, realistic, steady-programmed, programmable, IC-good, and Exec-preserving and has explicit jumps, no implicit jumps, non trivial ObjectKinds, and non trivial instruction set.

Let N be a set with non empty elements. Note that every definite non empty non void AMI over N which has non trivial ObjectKinds has also non trivial instruction set.

Let N be a set with non empty elements. One can check that every IC-Ins-separated non empty AMI over N which has non trivial ObjectKinds has also non trivial instruction locations.

Let N be a set with non empty elements, let A be a non empty AMI over N with non trivial ObjectKinds, and let o be an object of A. Observe that ObjectKind(o) is non trivial.

Let N be a set with non empty elements and let A be an AMI over N with non trivial instruction set. Note that the instructions of A is non trivial.

Let N be a set with non empty elements and let A be an IC-Ins-separated non empty AMI over N with non trivial instruction locations. Note that ObjectKind($\mathbf{IC}_A$) is non trivial.

Let N be a set with non empty elements, let A be a non empty non void AMI over N, and let I be an instruction of A. The functor Output I yielding a subset of the carrier of A is defined as follows:

(Def. 3) For every object o of A holds $o \in$ Output I iff there exists a state s of A such that $s(o) \neq (\mathrm{Exec}(I, s))(o)$.

Let N be a set with non empty elements, let A be an IC-Ins-separated definite non empty non void AMI over N, and let I be an instruction of A. The functor IODiff I yielding a subset of the carrier of A is defined by the condition (Def. 4).

(Def. 4) Let o be an object of A. Then $o \in$ IODiff I if and only if for every state s of A and for every element a of ObjectKind(o) holds $\mathrm{Exec}(I, s) = \mathrm{Exec}(I, s +\cdot (o, a))$.

The functor IOSum I yielding a subset of the carrier of A is defined by the condition (Def. 5).

(Def. 5) Let o be an object of A. Then $o \in$ IOSum I if and only if there exists a state s of A and there exists an element a of ObjectKind(o) such that $\mathrm{Exec}(I, s +\cdot (o, a)) \neq \mathrm{Exec}(I, s) +\cdot (o, a)$.

Let N be a set with non empty elements, let A be an IC-Ins-separated definite non empty non void AMI over N, and let I be an instruction of A. The functor Input I yielding a subset of the carrier of A is defined as follows:

(Def. 6) Input I = IOSum $I \setminus$ IODiff I.

The following propositions are true:

(9) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. Then IODiff I misses Input I.

(10) Let A be an IC-Ins-separated definite non empty non void AMI over N with non trivial ObjectKinds and I be an instruction of A. Then IODiff $I \subseteq$ Output I.

(11) For every IC-Ins-separated definite non empty non void AMI A over N and for every instruction I of A holds Output $I \subseteq$ IOSum I.

(12) For every IC-Ins-separated definite non empty non void AMI A over N and for every instruction I of A holds Input $I \subseteq$ IOSum I.

(13) Let A be an IC-Ins-separated definite non empty non void AMI over N with non trivial ObjectKinds and I be an instruction of A. Then $\mathrm{IODiff}\, I = \mathrm{Output}\, I \setminus \mathrm{Input}\, I$.

(14) Let A be an IC-Ins-separated definite non empty non void AMI over N with non trivial ObjectKinds and I be an instruction of A. Then $\mathrm{IOSum}\, I = \mathrm{Output}\, I \cup \mathrm{Input}\, I$.

(15) Let A be an IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, and o be an object of A. If ObjectKind(o) is trivial, then $o \notin \mathrm{IOSum}\, I$.

(16) Let A be an IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, and o be an object of A. If ObjectKind(o) is trivial, then $o \notin \mathrm{Input}\, I$.

(17) Let A be an IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, and o be an object of A. If ObjectKind(o) is trivial, then $o \notin \mathrm{Output}\, I$.

(18) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. Then I is halting if and only if $\mathrm{Output}\, I$ is empty.

(19) Let A be an IC-Ins-separated definite non empty non void AMI over N with non trivial ObjectKinds and I be an instruction of A. If I is halting, then $\mathrm{IODiff}\, I$ is empty.

(20) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If I is halting, then $\mathrm{IOSum}\, I$ is empty.

(21) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If I is halting, then $\mathrm{Input}\, I$ is empty.

Let N be a set with non empty elements, let A be a halting IC-Ins-separated definite non empty non void AMI over N, and let I be a halting instruction of A. One can verify the following observations:

* $\mathrm{Input}\, I$ is empty,
* $\mathrm{Output}\, I$ is empty, and
* $\mathrm{IOSum}\, I$ is empty.

Let N be a set with non empty elements, let A be a halting IC-Ins-separated definite non empty non void AMI over N with non trivial ObjectKinds, and let I be a halting instruction of A. Note that $\mathrm{IODiff}\, I$ is empty.

The following propositions are true:

(22) Let A be a steady-programmed IC-Ins-separated definite non empty non void AMI over N with non trivial instruction set, f be an instruction-location of A, and I be an instruction of A. Then $f \notin \mathrm{IODiff}\, I$.

(23) Let A be a standard IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If I is sequential, then $\mathbf{IC}_A \notin \mathrm{IODiff}\, I$.

(24) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If there exists a state s of A such that $(\mathrm{Exec}(I,s))(\mathbf{IC}_A) \neq \mathbf{IC}_s$, then $\mathbf{IC}_A \in \mathrm{Output}\, I$.

(25) Let A be a standard IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If I is sequential, then $\mathbf{IC}_A \in \mathrm{Output}\, I$.

(26) Let A be an IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If there exists a state s of A such that $(\mathrm{Exec}(I,s))(\mathbf{IC}_A) \neq \mathbf{IC}_s$, then $\mathbf{IC}_A \in \mathrm{IOSum}\, I$.

(27) Let A be a standard IC-Ins-separated definite non empty non void AMI over N and I be an instruction of A. If I is sequential, then $\mathbf{IC}_A \in \mathrm{IOSum}\, I$.

(28) Let A be an IC-Ins-separated definite non empty non void AMI over N, f be an instruction-location of A, and I be an instruction of A. Suppose that for every state s of A and for every programmed finite partial state p of A holds $\mathrm{Exec}(I, s{+}\!\cdot p) = \mathrm{Exec}(I,s){+}\!\cdot p$. Then $f \notin \mathrm{IOSum}\, I$.

(29) Let A be an IC-Ins-separated definite non empty non void AMI over N, I be an instruction of A, and o be an object of A. If I is jump-only, then if $o \in \mathrm{Output}\, I$, then $o = \mathbf{IC}_A$.

3. Input and Output of the Instructions of **SCM**

In the sequel a, b are data-locations, f is an instruction-location of **SCM**, and I is an instruction of **SCM**.

We now state two propositions:

(30) For every state s of **SCM** and for every element w of $\mathrm{ObjectKind}(\mathbf{IC}_{\mathbf{SCM}})$ holds $(s +\cdot (\mathbf{IC}_{\mathbf{SCM}}, w))(a) = s(a)$.

(31) $f \neq \mathrm{Next}(f)$.

Let s be a state of **SCM**, let d_1 be a data-location, and let k be an integer. Then $s +\cdot (d_1, k)$ is a state of **SCM**.

Let us observe that **SCM** has non trivial ObjectKinds.

Next we state a number of propositions:

(32) $\mathrm{IODiff}(a{:=}a) = \emptyset$.

(33) If $a \neq b$, then $\mathrm{IODiff}(a{:=}b) = \{a\}$.

(34) $\mathrm{IODiff}\, \mathrm{AddTo}(a,b) = \emptyset$.

(35) $\mathrm{IODiff}\, \mathrm{SubFrom}(a,a) = \{a\}$.

(36) If $a \neq b$, then $\mathrm{IODiff}\, \mathrm{SubFrom}(a,b) = \emptyset$.

(37) $\mathrm{IODiff}\, \mathrm{MultBy}(a,b) = \emptyset$.

(38) $\mathrm{IODiff\,Divide}(a,a) = \{a\}$.

(39) If $a \neq b$, then $\mathrm{IODiff\,Divide}(a,b) = \emptyset$.

(40) $\mathrm{IODiff\,goto}\ f = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(41) $\mathrm{IODiff}(\mathbf{if}\ a = 0\ \mathbf{goto}\ f) = \emptyset$.

(42) $\mathrm{IODiff}(\mathbf{if}\ a > 0\ \mathbf{goto}\ f) = \emptyset$.

(43) $\mathrm{Output}(a{:=}a) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(44) If $a \neq b$, then $\mathrm{Output}(a{:=}b) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(45) $\mathrm{Output\,AddTo}(a,b) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(46) $\mathrm{Output\,SubFrom}(a,b) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(47) $\mathrm{Output\,MultBy}(a,b) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(48) $\mathrm{Output\,Divide}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(49) $\mathrm{Output\,goto}\ f = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(50) $\mathrm{Output}(\mathbf{if}\ a = 0\ \mathbf{goto}\ f) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(51) $\mathrm{Output}(\mathbf{if}\ a > 0\ \mathbf{goto}\ f) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(52) $f \notin \mathrm{IOSum}\,I$.

(53) $\mathrm{IOSum}(a{:=}a) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(54) If $a \neq b$, then $\mathrm{IOSum}(a{:=}b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(55) $\mathrm{IOSum\,AddTo}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(56) $\mathrm{IOSum\,SubFrom}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(57) $\mathrm{IOSum\,MultBy}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(58) $\mathrm{IOSum\,Divide}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(59) $\mathrm{IOSum\,goto}\ f = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(60) $\mathrm{IOSum}(\mathbf{if}\ a = 0\ \mathbf{goto}\ f) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(61) $\mathrm{IOSum}(\mathbf{if}\ a > 0\ \mathbf{goto}\ f) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(62) $\mathrm{Input}(a{:=}a) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(63) If $a \neq b$, then $\mathrm{Input}(a{:=}b) = \{b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(64) $\mathrm{Input\,AddTo}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(65) $\mathrm{Input\,SubFrom}(a,a) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(66) If $a \neq b$, then $\mathrm{Input\,SubFrom}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(67) $\mathrm{Input\,MultBy}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(68) $\mathrm{Input\,Divide}(a,a) = \{\mathbf{IC}_{\mathbf{SCM}}\}$.

(69) If $a \neq b$, then $\mathrm{Input\,Divide}(a,b) = \{a, b, \mathbf{IC}_{\mathbf{SCM}}\}$.

(70) $\mathrm{Input\,goto}\ f = \emptyset$.

(71) $\mathrm{Input}(\mathbf{if}\ a = 0\ \mathbf{goto}\ f) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

(72) $\mathrm{Input}(\mathbf{if}\ a > 0\ \mathbf{goto}\ f) = \{a, \mathbf{IC}_{\mathbf{SCM}}\}$.

References

[1] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.
[2] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.
[3] Józef Białas. Group and field definitions. *Formalized Mathematics*, 1(**3**):433–439, 1990.
[4] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[5] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[6] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[7] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.
[8] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[9] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[10] Andrzej Trybulec, Piotr Rudnicki, and Artur Korniłowicz. Standard ordering of instruction locations. *Formalized Mathematics*, 9(**2**):291–301, 2001.
[11] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[12] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[13] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.

Received May 8, 2001