JOURNAL OF FORMALIZED MATHEMATICS
Volume 13, Released 2001, Published 2003
Inst. of Computer Science, Univ. of Białystok

Input and Output of Instructions[1]

Artur Korniłowicz
University of Białystok

MML Identifier: AMI_7.

WWW: http://mizar.org/JFM/Vol13/ami_7.html

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

1. Preliminaries

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

The following three propositions are true:

(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.

One can prove the following 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 $(\mathrm{Exec}(I,s))(f) = (\mathrm{Exec}(I,s+\cdot(o,w)))(f)$.

(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 $\mathbf{IC}_{\mathrm{Exec}(I,s+\cdot(o,w))} = \mathbf{IC}_{\mathrm{Exec}(I,s)+\cdot(o,w)}$.

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

(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 check that STC(N) has non trivial ObjectKinds.

Let N be a set with non empty elements. Note 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. Note 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. One can check 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. One can verify 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 A is defined by:

(Def. 3) For every object o of A holds $o \in \mathrm{Output}I$ 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 A is defined by the condition (Def. 4).

(Def. 4) Let o be an object of A. Then $o \in \mathrm{IODiff}I$ 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 A is defined by the condition (Def. 5).

(Def. 5) Let o be an object of A. Then $o \in \mathrm{IOSum}I$ 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 A is defined by:

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

We now state a number of propositions:

(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 $\mathrm{IODiff}\,I \subseteq \mathrm{Output}\,I$.

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

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

(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 $\mathrm{ObjectKind}(o)$ 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 $\mathrm{ObjectKind}(o)$ 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 $\mathrm{ObjectKind}(o)$ 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 check 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. One can check that $\mathrm{IODiff}\,I$ is empty.

Next we state several propositions:

(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**.

The following two propositions are true:

(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.

One can prove the following 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) $\mathrm{IODiff}\,\mathrm{Divide}(a,a) = \{a\}$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(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 and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/finseq_1.html`.

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

[3] Józef Białas. Group and field definitions. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/realset1.html`.

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

[5] Czesław Byliński. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/cqc_lang.html`.

[6] 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`.

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

[8] 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.

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

[10] Andrzej Trybulec. Binary operations applied to functions. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funcop_1.html.

[11] Andrzej Trybulec. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/enumset1.html.

[12] Andrzej Trybulec. Tarski Grothendieck set theory. *Journal of Formalized Mathematics*, Axiomatics, 1989. http://mizar.org/JFM/Axiomatics/tarski.html.

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

[14] 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.

[15] Andrzej Trybulec, Piotr Rudnicki, and Artur Korniłowicz. Standard ordering of instruction locations. *Journal of Formalized Mathematics*, 12, 2000. http://mizar.org/JFM/Vol12/amistd_1.html.

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

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

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

Received May 8, 2001

Published January 2, 2004