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

On the Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$[1]

Artur Korniłowicz
University of Białystok

MML Identifier: SCMFSA10.

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

For simplicity, we use the following convention: a, b are integer locations, f is a finite sequence location, i_1, i_2, i_3 are instruction-locations of $\mathbf{SCM}_{\mathrm{FSA}}$, T is an instruction type of $\mathbf{SCM}_{\mathrm{FSA}}$, and k is a natural number.

Next we state two propositions:

(1) For every function f and for all sets a, A, b, B, c, C such that $a \neq b$ and $a \neq c$ holds $(f+\cdot(a \mapsto A)+\cdot(b \mapsto B)+\cdot(c \mapsto C))(a) = A$.

(2) For all sets a, b holds $\langle a \rangle +\cdot (1, b) = \langle b \rangle$.

Let l_1, l_2 be integer locations and let a, b be integers. Then $[l_1 \longmapsto a, l_2 \longmapsto b]$ is a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$.

One can prove the following propositions:

(3) $a \notin$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(4) $f \notin$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(5) Data-$\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}} \neq$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(6) Data*-$\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}} \neq$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(7) Let o be an object of $\mathbf{SCM}_{\mathrm{FSA}}$. Then

(i) $o = \mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}$, or

(ii) $o \in$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$, or

(iii) o is an integer location or a finite sequence location.

(8) If $i_2 \neq i_3$, then $\mathrm{Next}(i_2) \neq \mathrm{Next}(i_3)$.

(9) $a{:=}b = \langle\!\langle 1, \langle a, b \rangle \rangle\!\rangle$.

(10) $\mathrm{AddTo}(a, b) = \langle\!\langle 2, \langle a, b \rangle \rangle\!\rangle$.

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

ISSN 1426–2630

(11) $\mathrm{SubFrom}(a, b) = \langle 3, \langle a, b\rangle\rangle$.

(12) $\mathrm{MultBy}(a, b) = \langle 4, \langle a, b\rangle\rangle$.

(13) $\mathrm{Divide}(a, b) = \langle 5, \langle a, b\rangle\rangle$.

(14) goto $i_1 = \langle 6, \langle i_1\rangle\rangle$.

(15) **if** $a = 0$ **goto** $i_1 = \langle 7, \langle i_1, a\rangle\rangle$.

(16) **if** $a > 0$ **goto** $i_1 = \langle 8, \langle i_1, a\rangle\rangle$.

(17) $\mathrm{AddressPart}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \emptyset$.

(18) $\mathrm{AddressPart}(a{:=}b) = \langle a, b\rangle$.

(19) $\mathrm{AddressPart}(\mathrm{AddTo}(a, b)) = \langle a, b\rangle$.

(20) $\mathrm{AddressPart}(\mathrm{SubFrom}(a, b)) = \langle a, b\rangle$.

(21) $\mathrm{AddressPart}(\mathrm{MultBy}(a, b)) = \langle a, b\rangle$.

(22) $\mathrm{AddressPart}(\mathrm{Divide}(a, b)) = \langle a, b\rangle$.

(23) $\mathrm{AddressPart}(\text{goto } i_2) = \langle i_2\rangle$.

(24) $\mathrm{AddressPart}(\textbf{if } a = 0 \textbf{ goto } i_2) = \langle i_2, a\rangle$.

(25) $\mathrm{AddressPart}(\textbf{if } a > 0 \textbf{ goto } i_2) = \langle i_2, a\rangle$.

(26) $\mathrm{AddressPart}(b{:=}f_a) = \langle b, f, a\rangle$.

(27) $\mathrm{AddressPart}(f_a{:=}b) = \langle b, f, a\rangle$.

(28) $\mathrm{AddressPart}(a{:=}\mathrm{len}f) = \langle a, f\rangle$.

(29) $\mathrm{AddressPart}(f{:=}\langle \underbrace{0, \ldots, 0}_{a}\rangle) = \langle a, f\rangle$.

(30) If $T = 0$, then $\mathrm{AddressParts}\, T = \{0\}$.

Let us consider T. Observe that $\mathrm{AddressParts}\, T$ is non empty.

Next we state a number of propositions:

(31) If $T = 1$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(32) If $T = 2$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(33) If $T = 3$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(34) If $T = 4$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(35) If $T = 5$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(36) If $T = 6$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1\}$.

(37) If $T = 7$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(38) If $T = 8$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(39) If $T = 9$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2, 3\}$.

(40) If $T = 10$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2, 3\}$.

(41) If $T = 11$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(42) If $T = 12$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1, 2\}$.

(43) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(a:=b)}(1) = \text{Data-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(44) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(a:=b)}(2) = \text{Data-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(45) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{AddTo}(a,b))}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(46) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{AddTo}(a,b))}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(47) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{SubFrom}(a,b))}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(48) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{SubFrom}(a,b))}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(49) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{MultBy}(a,b))}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(50) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{MultBy}(a,b))}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(51) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{Divide}(a,b))}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(52) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathrm{Divide}(a,b))}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(53) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathbf{goto}\ i_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(54) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathbf{if}\ a{=}0\ \mathbf{goto}\ i_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(55) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathbf{if}\ a{=}0\ \mathbf{goto}\ i_2)}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(56) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathbf{if}\ a{>}0\ \mathbf{goto}\ i_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(57) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(\mathbf{if}\ a{>}0\ \mathbf{goto}\ i_2)}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(58) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(b:=f_a)}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(59) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(b:=f_a)}(2) = \mathrm{Data}^*\text{-}\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(60) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(b:=f_a)}(3) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(61) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(f_a:=b)}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(62) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(f_a:=b)}(2) = \mathrm{Data}^*\text{-}\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(63) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(f_a:=b)}(3) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(64) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(a:=\mathrm{len}f)}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(65) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(a:=\mathrm{len}f)}(2) = \mathrm{Data}^*\text{-}\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(66) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(f:=\langle\underbrace{0,\ldots,0}_{a}\rangle)}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(67) $\prod_{\mathrm{AddressParts}\,\mathrm{InsCode}(f:=\langle\underbrace{0,\ldots,0}_{a}\rangle)}(2) = \mathrm{Data}^*\text{-}\mathrm{Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(68) $\mathrm{NIC}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}, i_1) = \{i_1\}$.

One can verify that $\mathrm{JUMP}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}})$ is empty.

We now state the proposition

(69) $\mathrm{NIC}(a{:=}b, i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. Note that $\mathrm{JUMP}(a{:=}b)$ is empty.

One can prove the following proposition

(70) $\mathrm{NIC}(\mathrm{AddTo}(a,b), i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. Note that $\mathrm{JUMP}(\mathrm{AddTo}(a,b))$ is empty.

Next we state the proposition

(71) $\mathrm{NIC}(\mathrm{SubFrom}(a,b), i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. Note that JUMP(SubFrom(a, b)) is empty.

One can prove the following proposition

(72) NIC(MultBy$(a, b), i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, b. Note that JUMP(MultBy(a, b)) is empty.

Next we state the proposition

(73) NIC(Divide$(a, b), i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, b. One can verify that JUMP(Divide(a, b)) is empty.

We now state two propositions:

(74) NIC(goto $i_2, i_1) = \{i_2\}$.

(75) JUMP(goto $i_2) = \{i_2\}$.

Let us consider i_2. One can verify that JUMP(goto i_2) is non empty and trivial.

We now state two propositions:

(76) $i_2 \in$ NIC(**if** $a = 0$ **goto** $i_2, i_1)$ and NIC(**if** $a = 0$ **goto** $i_2, i_1) \subseteq \{i_2, \text{Next}(i_1)\}$.

(77) JUMP(**if** $a = 0$ **goto** $i_2) = \{i_2\}$.

Let us consider a, i_2. One can check that JUMP(**if** $a = 0$ **goto** i_2) is non empty and trivial.

One can prove the following two propositions:

(78) $i_2 \in$ NIC(**if** $a > 0$ **goto** $i_2, i_1)$ and NIC(**if** $a > 0$ **goto** $i_2, i_1) \subseteq \{i_2, \text{Next}(i_1)\}$.

(79) JUMP(**if** $a > 0$ **goto** $i_2) = \{i_2\}$.

Let us consider a, i_2. Note that JUMP(**if** $a > 0$ **goto** i_2) is non empty and trivial.

The following proposition is true

(80) NIC(a:=$f_b, i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, b, f. Observe that JUMP(a:=f_b) is empty.

Next we state the proposition

(81) NIC(f_b:=$a, i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, b, f. One can check that JUMP(f_b:=a) is empty.

The following proposition is true

(82) NIC(a:=len$f, i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, f. Observe that JUMP(a:=len$f$) is empty.

The following proposition is true

(83) NIC(f:=$\langle\underbrace{0, \ldots, 0}_{a}\rangle, i_1) = \{\text{Next}(i_1)\}$.

Let us consider a, f. Note that JUMP(f:=$\langle\underbrace{0, \ldots, 0}_{a}\rangle$) is empty.

The following two propositions are true:

(84) $\mathrm{SUCC}(i_1) = \{i_1, \mathrm{Next}(i_1)\}$.

(85) Let f be a function from $\mathbb{N}$ into the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that for every natural number k holds $f(k) = \mathrm{insloc}(k)$. Then

(i) f is bijective, and

(ii) for every natural number k holds $f(k+1) \in \mathrm{SUCC}(f(k))$ and for every natural number j such that $f(j) \in \mathrm{SUCC}(f(k))$ holds $k \leqslant j$.

Let us observe that $\mathbf{SCM}_{\mathrm{FSA}}$ is standard.

The following propositions are true:

(86) $\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k) = \mathrm{insloc}(k)$.

(87) $\mathrm{Next}(\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k)) = \mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k+1)$.

(88) $\mathrm{Next}(i_1) = \mathrm{NextLoc}\, i_1$.

Let us mention that $\mathrm{InsCode}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}})$ is jump-only.

Let us mention that $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ is jump-only.

Let us consider i_2. One can verify that $\mathrm{InsCode}(\mathbf{goto}\ i_2)$ is jump-only.

Let us consider i_2. Observe that $\mathbf{goto}\ i_2$ is jump-only non sequential and non instruction location free.

Let us consider a, i_2. One can check that $\mathrm{InsCode}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2)$ is jump-only and $\mathrm{InsCode}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2)$ is jump-only.

Let us consider a, i_2. Observe that $\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2$ is jump-only non sequential and non instruction location free and $\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2$ is jump-only non sequential and non instruction location free.

Let us consider a, b. One can verify the following observations:

* $\mathrm{InsCode}(a{:=}b)$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{AddTo}(a, b))$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{SubFrom}(a, b))$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{MultBy}(a, b))$ is non jump-only, and
* $\mathrm{InsCode}(\mathrm{Divide}(a, b))$ is non jump-only.

Let us consider a, b. One can verify the following observations:

* $a{:=}b$ is non jump-only and sequential,
* $\mathrm{AddTo}(a, b)$ is non jump-only and sequential,
* $\mathrm{SubFrom}(a, b)$ is non jump-only and sequential,
* $\mathrm{MultBy}(a, b)$ is non jump-only and sequential, and
* $\mathrm{Divide}(a, b)$ is non jump-only and sequential.

Let us consider a, b, f. One can check that $\mathrm{InsCode}(b{:=}f_a)$ is non jump-only and $\mathrm{InsCode}(f_a{:=}b)$ is non jump-only.

Let us consider a, b, f. Observe that $b{:=}f_a$ is non jump-only and sequential and $f_a{:=}b$ is non jump-only and sequential.

Let us consider a, f. One can check that InsCode(a:=len$f$) is non jump-only and InsCode(f:=$\langle\underbrace{0,\ldots,0}_{a}\rangle$) is non jump-only.

Let us consider a, f. Note that a:=len$f$ is non jump-only and sequential and f:=$\langle\underbrace{0,\ldots,0}_{a}\rangle$ is non jump-only and sequential.

One can verify that $\mathbf{SCM}_{\mathrm{FSA}}$ is homogeneous and has explicit jumps and no implicit jumps.

Let us note that $\mathbf{SCM}_{\mathrm{FSA}}$ is regular.

The following propositions are true:

(89) IncAddr(goto i_2, k) = goto $\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2) + k)$.

(90) IncAddr(**if** $a = 0$ **goto** i_2, k) = **if** $a = 0$ **goto** $\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2) + k)$.

(91) IncAddr(**if** $a > 0$ **goto** i_2, k) = **if** $a > 0$ **goto** $\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2) + k)$.

Let us note that $\mathbf{SCM}_{\mathrm{FSA}}$ is IC-good and Exec-preserving.

References

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.

[2] Grzegorz Bancerek. Sequences of ordinal numbers. *Formalized Mathematics*, 1(**2**):281–290, 1990.

[3] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.

[4] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.

[5] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.

[6] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.

[7] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.

[8] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.

[9] 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.

[10] Artur Korniłowicz. On the composition of macro instructions of standard computers. *Formalized Mathematics*, 9(**2**):303–316, 2001.

[11] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.

[12] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.

[13] Yozo Toda. The formalization of simple graphs. *Formalized Mathematics*, 5(**1**):137–144, 1996.

[14] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.

[15] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.

[16] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(**4**):519–528, 1996.

[17] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **scm**. *Formalized Mathematics*, 5(**4**):507–512, 1996.

[18] Andrzej Trybulec, Piotr Rudnicki, and Artur Korniłowicz. Standard ordering of instruction locations. *Formalized Mathematics*, 9(**2**):291–301, 2001.
[19] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[20] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[21] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[22] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[23] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[24] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received May 8, 2001