JOURNAL OF FORMALIZED MATHEMATICS
Volume 8, Released 1996, Published 2003
Inst. of Computer Science, Univ. of Białystok

Memory Handling for $\mathbf{SCM}_{\rm FSA}$[1]

Piotr Rudnicki
University of Alberta
Edmonton

Andrzej Trybulec
Warsaw University
Białystok

Summary. We introduce some terminology for reasoning about memory used in programs in general and in macro instructions (introduced in [23]) in particular. The usage of integer locations and of finite sequence locations by a program is treated separately. We define some functors for selecting memory locations needed for local (temporary) variables in macro instructions. Some semantic properties of the introduced notions are given in terms of executions of macro instructions.

MML Identifier: SF_MASTR.

WWW: http://mizar.org/JFM/Vol8/sf_mastr.html

The articles [17], [15], [6], [27], [18], [10], [19], [25], [26], [16], [7], [11], [1], [13], [2], [8], [28], [4], [5], [9], [3], [12], [20], [14], [24], [21], [22], and [23] provide the notation and terminology for this paper.

1. Preliminaries

One can prove the following propositions:

(1) For all sets x, y, a and for every function f such that $f(x) = f(y)$ holds $f(a) = (f \cdot (\mathrm{id}_{\mathrm{dom}\, f} +\cdot (x,y)))(a)$.

(2) For all sets x, y and for every function f such that if $x \in \mathrm{dom}\, f$, then $y \in \mathrm{dom}\, f$ and $f(x) = f(y)$ holds $f = f \cdot (\mathrm{id}_{\mathrm{dom}\, f} +\cdot (x,y))$.

Let A be a finite set and let B be a set. One can check that every function from A into B is finite.

Let A be a finite set, let B be a set, and let f be a function from A into $\mathrm{Fin}\, B$. One can check that $\bigcup f$ is finite.

Let us mention that Int-Locations is non empty.

Let us observe that FinSeq-Locations is non empty.

2. Uniqueness of instruction components

For simplicity, we use the following convention: a, b, c, a_1, a_2, b_1, b_2 denote integer locations, l, l_1, l_2 denote instruction-locations of $\mathbf{SCM}_{\rm FSA}$, f, f_1, f_2 denote finite sequence locations, and i, j denote instructions of $\mathbf{SCM}_{\rm FSA}$.

We now state a number of propositions:

(5)[1] If $a_1 {:=} b_1 = a_2 {:=} b_2$, then $a_1 = a_2$ and $b_1 = b_2$.

[1] This work was partially supported by NSERC Grant OGP9207 and NATO CRG 951368.

[1] The propositions (3) and (4) have been removed.

(6) If $\mathrm{AddTo}(a_1,b_1) = \mathrm{AddTo}(a_2,b_2)$, then $a_1 = a_2$ and $b_1 = b_2$.

(7) If $\mathrm{SubFrom}(a_1,b_1) = \mathrm{SubFrom}(a_2,b_2)$, then $a_1 = a_2$ and $b_1 = b_2$.

(8) If $\mathrm{MultBy}(a_1,b_1) = \mathrm{MultBy}(a_2,b_2)$, then $a_1 = a_2$ and $b_1 = b_2$.

(9) If $\mathrm{Divide}(a_1,b_1) = \mathrm{Divide}(a_2,b_2)$, then $a_1 = a_2$ and $b_1 = b_2$.

(10) If goto $l_1 =$ goto l_2, then $l_1 = l_2$.

(11) If **if** $a_1 = 0$ **goto** $l_1 =$ **if** $a_2 = 0$ **goto** l_2, then $a_1 = a_2$ and $l_1 = l_2$.

(12) If **if** $a_1 > 0$ **goto** $l_1 =$ **if** $a_2 > 0$ **goto** l_2, then $a_1 = a_2$ and $l_1 = l_2$.

(13) If $b_1{:=}{f_1}_{a_1} = b_2{:=}{f_2}_{a_2}$, then $a_1 = a_2$ and $b_1 = b_2$ and $f_1 = f_2$.

(14) If ${f_1}_{a_1}{:=}b_1 = {f_2}_{a_2}{:=}b_2$, then $a_1 = a_2$ and $b_1 = b_2$ and $f_1 = f_2$.

(15) If $a_1{:=}\mathrm{len}\, f_1 = a_2{:=}\mathrm{len}\, f_2$, then $a_1 = a_2$ and $f_1 = f_2$.

(16) If $f_1{:=}\langle\underbrace{0,\ldots,0}_{a_1}\rangle = f_2{:=}\langle\underbrace{0,\ldots,0}_{a_2}\rangle$, then $a_1 = a_2$ and $f_1 = f_2$.

3. Integer locations used in macros

Let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{UsedIntLoc}(i)$ yielding an element of Fin Int-Locations is defined as follows:

(Def. 1)(i) There exist integer locations a, b such that $i = a{:=}b$ or $i = \mathrm{AddTo}(a,b)$ or $i = \mathrm{SubFrom}(a,b)$ or $i = \mathrm{MultBy}(a,b)$ or $i = \mathrm{Divide}(a,b)$ but $\mathrm{UsedIntLoc}(i) = \{a,b\}$ if $\mathrm{InsCode}(i) \in \{1,2,3,4,5\}$,

(ii) there exists an integer location a and there exists an instruction-location l of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $i =$ **if** $a = 0$ **goto** l or $i =$ **if** $a > 0$ **goto** l but $\mathrm{UsedIntLoc}(i) = \{a\}$ if $\mathrm{InsCode}(i) = 7$ or $\mathrm{InsCode}(i) = 8$,

(iii) there exist integer locations a, b and there exists a finite sequence location f such that $i = b{:=}f_a$ or $i = f_a{:=}b$ but $\mathrm{UsedIntLoc}(i) = \{a,b\}$ if $\mathrm{InsCode}(i) = 9$ or $\mathrm{InsCode}(i) = 10$,

(iv) there exists an integer location a and there exists a finite sequence location f such that $i = a{:=}\mathrm{len}\, f$ or $i = f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$ but $\mathrm{UsedIntLoc}(i) = \{a\}$ if $\mathrm{InsCode}(i) = 11$ or $\mathrm{InsCode}(i) = 12$,

(v) $\mathrm{UsedIntLoc}(i) = \emptyset$, otherwise.

The following propositions are true:

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

(18) If $i = a{:=}b$ or $i = \mathrm{AddTo}(a,b)$ or $i = \mathrm{SubFrom}(a,b)$ or $i = \mathrm{MultBy}(a,b)$ or $i = \mathrm{Divide}(a,b)$, then $\mathrm{UsedIntLoc}(i) = \{a,b\}$.

(19) $\mathrm{UsedIntLoc}(\mathrm{goto}\ l) = \emptyset$.

(20) If $i =$ **if** $a = 0$ **goto** l or $i =$ **if** $a > 0$ **goto** l, then $\mathrm{UsedIntLoc}(i) = \{a\}$.

(21) If $i = b{:=}f_a$ or $i = f_a{:=}b$, then $\mathrm{UsedIntLoc}(i) = \{a,b\}$.

(22) If $i = a{:=}\mathrm{len}\, f$ or $i = f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$, then $\mathrm{UsedIntLoc}(i) = \{a\}$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{UsedIntLoc}(p)$ yielding a subset of Int-Locations is defined by the condition (Def. 2).

(Def. 2) There exists a function U_1 from the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ into Fin Int-Locations such that for every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $U_1(i) = \mathrm{UsedIntLoc}(i)$ and $\mathrm{UsedIntLoc}(p) = \bigcup(U_1 \cdot p)$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. One can check that $\mathrm{UsedIntLoc}(p)$ is finite.

We adopt the following convention: p, r are programmed finite partial states of $\mathbf{SCM}_{\mathrm{FSA}}$, I, J are macro instructions, and k, m, n are natural numbers.

We now state a number of propositions:

(23) If $i \in \mathrm{rng}\, p$, then $\mathrm{UsedIntLoc}(i) \subseteq \mathrm{UsedIntLoc}(p)$.

(24) $\mathrm{UsedIntLoc}(p {+\cdot} r) \subseteq \mathrm{UsedIntLoc}(p) \cup \mathrm{UsedIntLoc}(r)$.

(25) If $\mathrm{dom}\, p$ misses $\mathrm{dom}\, r$, then $\mathrm{UsedIntLoc}(p {+\cdot} r) = \mathrm{UsedIntLoc}(p) \cup \mathrm{UsedIntLoc}(r)$.

(26) $\mathrm{UsedIntLoc}(p) = \mathrm{UsedIntLoc}(\mathrm{Shift}(p,k))$.

(27) $\mathrm{UsedIntLoc}(i) = \mathrm{UsedIntLoc}(\mathrm{IncAddr}(i,k))$.

(28) $\mathrm{UsedIntLoc}(p) = \mathrm{UsedIntLoc}(\mathrm{IncAddr}(p,k))$.

(29) $\mathrm{UsedIntLoc}(I) = \mathrm{UsedIntLoc}(\mathrm{ProgramPart}(\mathrm{Relocated}(I,k)))$.

(30) $\mathrm{UsedIntLoc}(I) = \mathrm{UsedIntLoc}(\mathrm{Directed}(I))$.

(31) $\mathrm{UsedIntLoc}(I;\, J) = \mathrm{UsedIntLoc}(I) \cup \mathrm{UsedIntLoc}(J)$.

(32) $\mathrm{UsedIntLoc}(\mathrm{Macro}(i)) = \mathrm{UsedIntLoc}(i)$.

(33) $\mathrm{UsedIntLoc}(i;\, J) = \mathrm{UsedIntLoc}(i) \cup \mathrm{UsedIntLoc}(J)$.

(34) $\mathrm{UsedIntLoc}(I;\, j) = \mathrm{UsedIntLoc}(I) \cup \mathrm{UsedIntLoc}(j)$.

(35) $\mathrm{UsedIntLoc}(i;\, j) = \mathrm{UsedIntLoc}(i) \cup \mathrm{UsedIntLoc}(j)$.

4. FINITE SEQUENCE LOCATIONS USED IN MACROS

Let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i)$ yielding an element of Fin FinSeq-Locations is defined as follows:

(Def. 3)(i) There exist integer locations a, b and there exists a finite sequence location f such that $i = b{:=}f_a$ or $i = f_a{:=}b$ but $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \{f\}$ if $\mathrm{InsCode}(i) = 9$ or $\mathrm{InsCode}(i) = 10$,

(ii) there exists an integer location a and there exists a finite sequence location f such that $i = a{:=}\mathrm{len}\, f$ or $i = f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$ but $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \{f\}$ if $\mathrm{InsCode}(i) = 11$ or $\mathrm{InsCode}(i) = 12$,

(iii) $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \emptyset$, otherwise.

The following propositions are true:

(36) If $i = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ or $i = a{:=}b$ or $i = \mathrm{AddTo}(a,b)$ or $i = \mathrm{SubFrom}(a,b)$ or $i = \mathrm{MultBy}(a,b)$ or $i = \mathrm{Divide}(a,b)$ or $i = \mathrm{goto}\, l$ or $i = \mathbf{if}\ a = 0\ \mathbf{goto}\ l$ or $i = \mathbf{if}\ a > 0\ \mathbf{goto}\ l$, then $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \emptyset$.

(37) If $i = b{:=}f_a$ or $i = f_a{:=}b$, then $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \{f\}$.

(38) If $i = a{:=}\mathrm{len}\, f$ or $i = f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$, then $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \{f\}$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{UsedInt}^{*}\,\mathrm{Loc}(p)$ yielding a subset of FinSeq-Locations is defined by the condition (Def. 4).

(Def. 4) There exists a function U_1 from the instructions of $\mathbf{SCM}_{\rm FSA}$ into Fin FinSeq-Locations such that for every instruction i of $\mathbf{SCM}_{\rm FSA}$ holds $U_1(i) = \text{UsedInt}^*\,\text{Loc}(i)$ and $\text{UsedInt}^*\,\text{Loc}(p) = \bigcup(U_1 \cdot p)$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\rm FSA}$. Note that $\text{UsedInt}^*\,\text{Loc}(p)$ is finite.
We now state a number of propositions:

(39) If $i \in \operatorname{rng} p$, then $\text{UsedInt}^*\,\text{Loc}(i) \subseteq \text{UsedInt}^*\,\text{Loc}(p)$.

(40) $\text{UsedInt}^*\,\text{Loc}(p{+}{\cdot}r) \subseteq \text{UsedInt}^*\,\text{Loc}(p) \cup \text{UsedInt}^*\,\text{Loc}(r)$.

(41) If $\operatorname{dom} p$ misses $\operatorname{dom} r$, then $\text{UsedInt}^*\,\text{Loc}(p{+}{\cdot}r) = \text{UsedInt}^*\,\text{Loc}(p) \cup \text{UsedInt}^*\,\text{Loc}(r)$.

(42) $\text{UsedInt}^*\,\text{Loc}(p) = \text{UsedInt}^*\,\text{Loc}(\text{Shift}(p,k))$.

(43) $\text{UsedInt}^*\,\text{Loc}(i) = \text{UsedInt}^*\,\text{Loc}(\text{IncAddr}(i,k))$.

(44) $\text{UsedInt}^*\,\text{Loc}(p) = \text{UsedInt}^*\,\text{Loc}(\text{IncAddr}(p,k))$.

(45) $\text{UsedInt}^*\,\text{Loc}(I) = \text{UsedInt}^*\,\text{Loc}(\text{ProgramPart}(\text{Relocated}(I,k)))$.

(46) $\text{UsedInt}^*\,\text{Loc}(I) = \text{UsedInt}^*\,\text{Loc}(\text{Directed}(I))$.

(47) $\text{UsedInt}^*\,\text{Loc}(I;\,J) = \text{UsedInt}^*\,\text{Loc}(I) \cup \text{UsedInt}^*\,\text{Loc}(J)$.

(48) $\text{UsedInt}^*\,\text{Loc}(\text{Macro}(i)) = \text{UsedInt}^*\,\text{Loc}(i)$.

(49) $\text{UsedInt}^*\,\text{Loc}(i;\,J) = \text{UsedInt}^*\,\text{Loc}(i) \cup \text{UsedInt}^*\,\text{Loc}(J)$.

(50) $\text{UsedInt}^*\,\text{Loc}(I;\,j) = \text{UsedInt}^*\,\text{Loc}(I) \cup \text{UsedInt}^*\,\text{Loc}(j)$.

(51) $\text{UsedInt}^*\,\text{Loc}(i;\,j) = \text{UsedInt}^*\,\text{Loc}(i) \cup \text{UsedInt}^*\,\text{Loc}(j)$.

5. Choosing an Integer Location not Used in a Program

Let I_1 be an integer location. We say that I_1 is read-only if and only if:

(Def. 5) $I_1 = \text{intloc}(0)$.

We introduce I_1 is read-write as an antonym of I_1 is read-only.
One can verify that $\text{intloc}(0)$ is read-only.
Let us note that there exists an integer location which is read-write.
In the sequel L denotes a finite subset of Int-Locations.
Let L be a finite subset of Int-Locations. The functor $\text{FirstNotIn}(L)$ yielding an integer location is defined as follows:

(Def. 6) There exists a non empty subset s_1 of $\mathbb{N}$ such that $\text{FirstNotIn}(L) = \text{intloc}(\min s_1)$ and $s_1 = \{k;k$ ranges over natural numbers: $\text{intloc}(k) \notin L\}$.

Next we state two propositions:

(52) $\text{FirstNotIn}(L) \notin L$.

(53) If $\text{FirstNotIn}(L) = \text{intloc}(m)$ and $\text{intloc}(n) \notin L$, then $m \leq n$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\rm FSA}$. The functor $\text{FirstNotUsed}(p)$ yielding an integer location is defined by:

(Def. 7) There exists a finite subset s_2 of Int-Locations such that $s_2 = \text{UsedIntLoc}(p) \cup \{\text{intloc}(0)\}$ and $\text{FirstNotUsed}(p) = \text{FirstNotIn}(s_2)$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\rm FSA}$. Note that $\text{FirstNotUsed}(p)$ is read-write.
One can prove the following propositions:

(54) FirstNotUsed$(p) \notin$ UsedIntLoc(p).

(55) If a:=$b \in \operatorname{rng} p$ or AddTo$(a,b) \in \operatorname{rng} p$ or SubFrom$(a,b) \in \operatorname{rng} p$ or MultBy$(a,b) \in \operatorname{rng} p$ or Divide$(a,b) \in \operatorname{rng} p$, then FirstNotUsed$(p) \neq a$ and FirstNotUsed$(p) \neq b$.

(56) If **if** $a = 0$ **goto** $l \in \operatorname{rng} p$ or **if** $a > 0$ **goto** $l \in \operatorname{rng} p$, then FirstNotUsed$(p) \neq a$.

(57) If b:=$f_a \in \operatorname{rng} p$ or f_a:=$b \in \operatorname{rng} p$, then FirstNotUsed$(p) \neq a$ and FirstNotUsed$(p) \neq b$.

(58) If a:=len$f \in \operatorname{rng} p$ or f:=$\langle\underbrace{0,\ldots,0}_{a}\rangle \in \operatorname{rng} p$, then FirstNotUsed$(p) \neq a$.

6. Choosing a finite sequence location not used in a program

In the sequel L is a finite subset of FinSeq-Locations.

Let L be a finite subset of FinSeq-Locations. The functor First* NotIn(L) yielding a finite sequence location is defined as follows:

(Def. 8) There exists a non empty subset s_1 of $\mathbb{N}$ such that First* NotIn(L) = fsloc$(\min s_1)$ and $s_1 = \{k; k$ ranges over natural numbers: fsloc$(k) \notin L\}$.

Next we state two propositions:

(59) First* NotIn$(L) \notin L$.

(60) If First* NotIn(L) = fsloc(m) and fsloc$(n) \notin L$, then $m \leq n$.

Let p be a programmed finite partial state of $\mathbf{SCM}_{\rm FSA}$. The functor First* NotUsed(p) yields a finite sequence location and is defined by:

(Def. 9) There exists a finite subset s_2 of FinSeq-Locations such that s_2 = UsedInt* Loc(p) and First* NotUsed(p) = First* NotIn(s_2).

Next we state three propositions:

(61) First* NotUsed$(p) \notin$ UsedInt* Loc(p).

(62) If b:=$f_a \in \operatorname{rng} p$ or f_a:=$b \in \operatorname{rng} p$, then First* NotUsed$(p) \neq f$.

(63) If a:=len$f \in \operatorname{rng} p$ or f:=$\langle\underbrace{0,\ldots,0}_{a}\rangle \in \operatorname{rng} p$, then First* NotUsed$(p) \neq f$.

7. Semantics

In the sequel s, t are states of $\mathbf{SCM}_{\rm FSA}$.

One can prove the following propositions:

(64) dom$I$ misses dom Start-At(insloc(n)).

(65) $\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}} \in$ dom$(I$+· Start-At(insloc$(n)))$.

(66) $(I$+· Start-At(insloc$(n)))(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}})$ = insloc(n).

(67) If I+· Start-At(insloc$(n)) \subseteq s$, then $\mathbf{IC}_s$ = insloc(n).

(68) If $c \notin$ UsedIntLoc(i), then (Exec$(i,s))(c) = s(c)$.

(69) If I+· Start-At(insloc$(0)) \subseteq s$ and for every m such that $m < n$ holds $\mathbf{IC}_{(\text{Computation}(s))(m)} \in$ dom$I$ and $a \notin$ UsedIntLoc(I), then (Computation$(s))(n)(a) = s(a)$.

(70) If $f \notin$ UsedInt* Loc(i), then (Exec$(i,s))(f) = s(f)$.

(71) If $I +\cdot$ Start-At(insloc(0)) $\subseteq s$ and for every m such that $m < n$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(m)} \in \mathrm{dom}\, I$ and $f \notin \mathrm{UsedInt}^{*}\,\mathrm{Loc}(I)$, then $(\mathrm{Computation}(s))(n)(f) = s(f)$.

(72) If $s \restriction \mathrm{UsedIntLoc}(i) = t \restriction \mathrm{UsedIntLoc}(i)$ and $s \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = t \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(i)$ and $\mathbf{IC}_{s} = \mathbf{IC}_{t}$, then $\mathbf{IC}_{\mathrm{Exec}(i,s)} = \mathbf{IC}_{\mathrm{Exec}(i,t)}$ and $\mathrm{Exec}(i,s) \restriction \mathrm{UsedIntLoc}(i) = \mathrm{Exec}(i,t) \restriction \mathrm{UsedIntLoc}(i)$ and $\mathrm{Exec}(i,s) \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(i) = \mathrm{Exec}(i,t) \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(i)$.

(73) Suppose $I +\cdot$ Start-At(insloc(0)) $\subseteq s$ and $I +\cdot$ Start-At(insloc(0)) $\subseteq t$ and $s \restriction \mathrm{UsedIntLoc}(I) = t \restriction \mathrm{UsedIntLoc}(I)$ and $s \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(I) = t \restriction \mathrm{UsedInt}^{*}\,\mathrm{Loc}(I)$ and for every m such that $m < n$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(m)} \in \mathrm{dom}\, I$. Then

(i) for every m such that $m < n$ holds $\mathbf{IC}_{(\mathrm{Computation}(t))(m)} \in \mathrm{dom}\, I$, and

(ii) for every m such that $m \leq n$ holds $\mathbf{IC}_{(\mathrm{Computation}(s))(m)} = \mathbf{IC}_{(\mathrm{Computation}(t))(m)}$ and for every a such that $a \in \mathrm{UsedIntLoc}(I)$ holds $(\mathrm{Computation}(s))(m)(a) = (\mathrm{Computation}(t))(m)(a)$ and for every f such that $f \in \mathrm{UsedInt}^{*}\,\mathrm{Loc}(I)$ holds $(\mathrm{Computation}(s))(m)(f) = (\mathrm{Computation}(t))(m)(f)$.

REFERENCES

[1] Grzegorz Bancerek. Cardinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/card_1.html.

[2] 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.

[3] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/funct_7.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. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funct_2.html.

[6] Czesław Byliński. Some basic properties of sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/zfmisc_1.html.

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

[8] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/finseq_2.html.

[9] 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.

[10] Agata Darmochwał. Finite sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finset_1.html.

[11] Agata Darmochwał and Andrzej Trybulec. Similarity of formulae. *Journal of Formalized Mathematics*, 3, 1991. http://mizar.org/JFM/Vol3/cqc_sim1.html.

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

[13] Andrzej Nędzusiak. σ-fields and probability. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/prob_1.html.

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

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

[16] Andrzej Trybulec. Semilattice operations on finite subsets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/setwiseo.html.

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

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

[19] Andrzej Trybulec and Agata Darmochwał. Boolean domains. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finsub_1.html.

[20] 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.

[21] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa_4.html.

[22] Andrzej Trybulec and Yatsuka Nakamura. Relocability for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa_5.html.

[23] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa6a.html.

[24] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/scmfsa_2.html.

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

[26] Wojciech A. Trybulec. Groups. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/group_1.html.

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

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

Received July 18, 1996

Published January 2, 2004