FORMALIZED MATHEMATICS
Number 1, January 1990
Université Catholique de Louvain

Binary Operations

Czesław Byliński[1]
Warsaw University
Białystok

Summary. In this paper we define binary and unary operations on domains. We also define the following predicates concerning the operations: ... is commutative, ... is associative, ... is the unity of ..., and ... is distributive wrt .... A number of schemes useful in justifying the existence of the operations are proved.

The articles [3], [1], and [2] provide the notation and terminology for this paper. The arguments of the notions defined below are the following: f which is an object of the type Function; a, b which are objects of the type Any. The functor

$$f\,.(a,b),$$

with values of the type Any, is defined by

$$\mathbf{it} = f\,.\langle a,b\rangle.$$

One can prove the following proposition

(1) **for** f **being** Function **for** a,b **being** Any **holds** $f\,.(a,b) = f\,.\langle a,b\rangle$.

In the sequel A, B, C will denote objects of the type DOMAIN. The arguments of the notions defined below are the following: A, B, C which are objects of the type reserved above; f which is an object of the type Function **of** $[:A,B:]$, C; a which is an object of the type Element **of** A; b which is an object of the type Element **of** B. Let us note that it makes sense to consider the following functor on a restricted area. Then

$$f\,.(a,b) \quad \text{is} \quad \text{Element } \mathbf{of}\ C.$$

The following proposition is true

(2) **for** $f1$,$f2$ **being** Function **of** $[:A,B:]$,C **st**
for a **being** Element **of** A

[1]Supported by RPBP.III-24.C1.

ISSN 0777-4028

$$\textbf{for}\ b\ \textbf{being}\ \text{Element}\ \textbf{of}\ B\ \textbf{holds}\ f1.(a,b) = f2.(a,b)$$
$$\textbf{holds}\ f1 = f2.$$

We now define two new modes. Let us consider A.

Unary_Operation **of** A stands for Function **of** A, A.

Binary_Operation **of** A stands for Function **of** $[:A, A:], A$.

We now state a proposition

(3) **for** f **being** Function **of** A, A **holds** f **is** Unary_Operation **of** A.

In the sequel u denotes an object of the type Unary_Operation **of** A. Next we state a proposition

(4) **for** f **being** Function **of** $[:A, A:], A$ **holds** f **is** Binary_Operation **of** A.

In the article we present several logical schemes. The scheme *UnOpEx* concerns a constant $\mathcal{A}$ that has the type DOMAIN and a binary predicate $\mathcal{P}$ and states that the following holds

ex u **being** Unary_Operation **of** $\mathcal{A}$ **st for** x **being** Element **of** $\mathcal{A}$ **holds** $\mathcal{P}[x, u.x]$

provided the parameters satisfy the following conditions:

- **for** x **being** Element **of** $\mathcal{A}$ **ex** y **being** Element **of** $\mathcal{A}$ **st** $\mathcal{P}[x, y]$,
- **for** x,$y1$,$y2$ **being** Element **of** $\mathcal{A}$ **st** $\mathcal{P}[x, y1]$ & $\mathcal{P}[x, y2]$ **holds** $y1 = y2$.

The scheme *UnOpLambda* concerns a constant $\mathcal{A}$ that has the type DOMAIN and a unary functor $\mathcal{F}$ yielding values of the type Element **of** $\mathcal{A}$ and states that the following holds

ex u **being** Unary_Operation **of** $\mathcal{A}$ **st for** x **being** Element **of** $\mathcal{A}$ **holds** $u.x = \mathcal{F}(x)$

for all values of the parameters.

For simplicity we adopt the following convention: o, o' will have the type Binary_Operation **of** A; a, b, c, e, $e1$, $e2$ will have the type Element **of** A. Let us consider A, o, a, b. Let us note that it makes sense to consider the following functor on a restricted area. Then

$o.(a, b)$ is Element **of** A.

Now we present two schemes. The scheme *BinOpEx* concerns a constant $\mathcal{A}$ that has the type DOMAIN and a ternary predicate $\mathcal{P}$ and states that the following holds

ex o **being** Binary_Operation **of** $\mathcal{A}$
st for a,b **being** Element **of** $\mathcal{A}$ **holds** $\mathcal{P}[a, b, o.(a, b)]$

provided the parameters satisfy the following conditions:

- **for** x,y **being** Element **of** $\mathcal{A}$ **ex** z **being** Element **of** $\mathcal{A}$ **st** $\mathcal{P}[x, y, z]$,

- **for** x,y **being** Element **of** $\mathcal{A}$
  for $z1$,$z2$ **being** Element **of** $\mathcal{A}$ **st** $\mathcal{P}[x, y, z1]$ & $\mathcal{P}[x, y, z2]$ **holds** $z1 = z2$.

The scheme *BinOpLambda* concerns a constant $\mathcal{A}$ that has the type DOMAIN and a binary functor $\mathcal{F}$ yielding values of the type Element **of** $\mathcal{A}$ and states that the following holds

ex o **being** Binary_Operation **of** $\mathcal{A}$
st for a,b **being** Element **of** $\mathcal{A}$ **holds** $o.(a, b) = \mathcal{F}(a, b)$

for all values of the parameters.

We now define three new predicates. Let us consider A, o. The predicate

o is_commutative is defined by **for** a,b **holds** $o.(a, b) = o.(b, a)$.

The predicate

o is_associative is defined by **for** a,b,c **holds** $o.(a, o.(b, c)) = o.(o.(a, b),c)$.

The predicate

o is_an_idempotentOp is defined by **for** a **holds** $o.(a, a) = a$.

Next we state three propositions:

(5) o is_commutative **iff for** a,b **holds** $o.(a, b) = o.(b, a)$,

(6) o is_associative **iff for** a,b,c **holds** $o.(a, o.(b, c)) = o.(o.(a, b),c)$,

(7) o is_an_idempotentOp **iff for** a **holds** $o.(a, a) = a$.

We now define two new predicates. Let us consider A, e, o. The predicate

e is_a_left_unity_wrt o is defined by **for** a **holds** $o.(e, a) = a$.

The predicate

e is_a_right_unity_wrt o is defined by **for** a **holds** $o.(a, e) = a$.

Let us consider A, e, o. The predicate

e is_a_unity_wrt o is defined by e is_a_left_unity_wrt o & e is_a_right_unity_wrt o.

We now state a number of propositions:

(8) e is_a_left_unity_wrt o **iff for** a **holds** $o.(e, a) = a$,

(9) e is_a_right_unity_wrt o **iff for** a **holds** $o.(a,e) = a$,

(10) e is_a_unity_wrt o **iff** e is_a_left_unity_wrt o & e is_a_right_unity_wrt o,

(11) e is_a_unity_wrt o **iff for** a **holds** $o.(e,a) = a$ & $o.(a,e) = a$,

(12) o is_commutative **implies** (e is_a_unity_wrt o **iff for** a **holds** $o.(e,a) = a$),

(13) o is_commutative **implies** (e is_a_unity_wrt o **iff for** a **holds** $o.(a,e) = a$),

(14) o is_commutative **implies** (e is_a_unity_wrt o **iff** e is_a_left_unity_wrt o),

(15) o is_commutative **implies** (e is_a_unity_wrt o **iff** e is_a_right_unity_wrt o),

(16) o is_commutative **implies** (e is_a_left_unity_wrt o **iff** e is_a_right_unity_wrt o),

(17) $e1$ is_a_left_unity_wrt o & $e2$ is_a_right_unity_wrt o **implies** $e1 = e2$,

(18) $e1$ is_a_unity_wrt o & $e2$ is_a_unity_wrt o **implies** $e1 = e2$.

Let us consider A, o. Assume that the following holds

ex e **st** e is_a_unity_wrt o.

The functor

the_unity_wrt o,

with values of the type Element **of** A, is defined by

it is_a_unity_wrt o.

One can prove the following proposition

(19) (**ex** e **st** e is_a_unity_wrt o)
implies for e **holds** $e =$ the_unity_wrt o **iff** e is_a_unity_wrt o.

We now define two new predicates. Let us consider A, o', o. The predicate

o' is_left_distributive_wrt o

is defined by

for a,b,c **holds** $o'.(a, o.(b,c)) = o.(o'.(a,b), o'.(a,c))$.

The predicate

o' is_right_distributive_wrt o

is defined by

for a,b,c **holds** $o'.(o.(a,b), c) = o.(o'.(a,c), o'.(b,c))$.

Let us consider A, o', o. The predicate

$$o' \text{ is\_distributive\_wrt } o$$

is defined by

$$o' \text{ is\_left\_distributive\_wrt } o \;\&\; o' \text{ is\_right\_distributive\_wrt } o.$$

We now state several propositions:

(20) $$o' \text{ is\_left\_distributive\_wrt } o \;\textbf{iff for } a,b,c \textbf{ holds } o'.(a, o.(b,c)) = o.(o'.(a,b), o'.(a,c)),$$

(21) $$o' \text{ is\_right\_distributive\_wrt } o \;\textbf{iff for } a,b,c \textbf{ holds } o'.(o.(a,b), c) = o.(o'.(a,c), o'.(b,c)),$$

(22) $$o' \text{ is\_distributive\_wrt } o \;\textbf{iff } o' \text{ is\_left\_distributive\_wrt } o \;\&\; o' \text{ is\_right\_distributive\_wrt } o,$$

(23) $$o' \text{ is\_distributive\_wrt } o \;\textbf{iff for } a,b,c \textbf{ holds } o'.(a, o.(b,c)) = o.(o'.(a,b), o'.(a,c)) \;\&\; o'.(o.(a,b), c) = o.(o'.(a,c), o'.(b,c)),$$

(24) $$o' \text{ is\_commutative } \textbf{implies } (o' \text{ is\_distributive\_wrt } o \;\textbf{iff for } a,b,c \textbf{ holds } o'.(a, o.(b,c)) = o.(o'.(a,b), o'.(a,c))),$$

(25) $$o' \text{ is\_commutative } \textbf{implies } (o' \text{ is\_distributive\_wrt } o \;\textbf{iff for } a,b,c \textbf{ holds } o'.(o.(a,b), c) = o.(o'.(a,c), o'.(b,c))),$$

(26) $$o' \text{ is\_commutative } \textbf{implies } (o' \text{ is\_distributive\_wrt } o \;\textbf{iff } o' \text{ is\_left\_distributive\_wrt } o),$$

(27) $$o' \text{ is\_commutative } \textbf{implies } (o' \text{ is\_distributive\_wrt } o \;\textbf{iff } o' \text{ is\_right\_distributive\_wrt } o),$$

(28) $$o' \text{ is\_commutative } \textbf{implies } (o' \text{ is\_right\_distributive\_wrt } o \;\textbf{iff } o' \text{ is\_left\_distributive\_wrt } o).$$

Let us consider A, u, o. The predicate

u is_distributive_wrt o is defined by **for** a,b **holds** $u.(o.(a,b)) = o.((u.a),(u.b))$.

The following proposition is true

(29) u is_distributive_wrt o **iff for** a,b **holds** $u.(o.(a,b)) = o.((u.a),(u.b))$.

References

[1] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1, 1990.

[2] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1, 1990.

[3] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1, 1990.

Received April 14, 1989