Fresh development of Zermelo 1908b in Lestrade

Randall Holmes

December 3, 2020

Consider this a proper lab notebook for development of Zermelo's approach to foundations under Lestrade, and also a diary of recovery from COVID-19.

1 Version notes

This subsection will have entries describing the development of the work

Dec 3, 2020: Starting. Setting myself the task.

The idea is to develop Zermelo 1908b, the paper on axiomatics of set theory, by directly reading the text. The first thing I need to decide is how to treat logical notions (do I have a preamble, or can one fold logical primitives into set theory primitives?)

2 Zermelo 1908b in Lestrade, with notes

The text that follows is organized by Zermelo's paragraph numbers.

1. The domain $\mathcal{B}$ of individuals will be represented by the built-in Lestrade type `obj`.

   The relation of equality must be declared. Do we declare inequality or do we declare negation?

   Dec 3 I am experimenting with inequality as a primitive: we will see what reasoning principles we need by following the text.

```
begin Lestrade execution

   >>> declare x obj


   x : obj


   {move 1}

   >>> declare y obj


   y : obj


   {move 1}

   >>> postulate = x y prop


   = : [(x_1 : obj), (y_1 : obj) =>
       (--- : prop)]


   {move 0}

   >>> postulate =/= x y prop


   =/= : [(x_1 : obj), (y_1 : obj) =>
       (--- : prop)]


   {move 0}
end Lestrade execution
```

2. The primitive relation of membership and the notion of being a set must be declared.

   The primitive function Isset witnesses that objects with elements are sets.

```
begin Lestrade execution

   >>> clearcurrent

{move 1}

   >>> declare a obj


   a : obj


   {move 1}

   >>> declare b obj


   b : obj


   {move 1}

   >>> postulate E a b prop


   E : [(a_1 : obj), (b_1 : obj) =>
       (--- : prop)]


   {move 0}
```

```
   >>> postulate set b prop


   set : [(b_1 : obj) => (--- : prop)]


   {move 0}

   >>> declare memberdata that a E b


   memberdata : that a E b


   {move 1}

   >>> postulate Isset memberdata that set \
       b


   Isset : [(.a_1 : obj), (.b_1 : obj), (memberdata_1
       : that .a_1 E .b_1) => (--- : that
       set (.b_1))]


   {move 0}
end Lestrade execution

% paragraph 3

\item  The subset relation (implication?)

begin Lestrade execution

   >>> declare M obj


   M : obj
```

```
{move 1}

>>> declare N obj


N : obj


{move 1}

>>> postulate << M N prop


<< : [(M_1 : obj), (N_1 : obj) =>
    (--- : prop)]


{move 0}

>>> declare subsetev that M << N


subsetev : that M << N


{move 1}

>>> postulate Subset1 subsetev that set \
    M


Subset1 : [(.M_1 : obj), (.N_1 : obj), (subsetev_1
    : that .M_1 << .N_1) => (--- : that
    set (.M_1))]
```

```
{move 0}

>>> postulate Subset2 subsetev that set \
    N


Subset2 : [(.M_1 : obj), (.N_1 : obj), (subsetev_1
    : that .M_1 << .N_1) => (--- : that
    set (.N_1))]


{move 0}

>>> declare x obj


x : obj


{move 1}

>>> declare memberev that x E M


memberev : that x E M


{move 1}

>>> postulate Subset3 subsetev memberev \
    that x E N


Subset3 : [(.M_1 : obj), (.N_1 : obj), (subsetev_1
    : that .M_1 << .N_1), (.x_1 : obj), (memberev_1
    : that .x_1 E .M_1) => (--- : that
    .x_1 E .N_1)]
```

```
   {move 0}

   >>> declare subsetev2 [x, memberev => \
          that x E N]


   subsetev2 : [(x_1 : obj), (memberev_1
       : that x_1 E M) => (--- : that x_1
       E N)]


   {move 1}

   >>> postulate Subset4 subsetev2 that M << \
       N


   Subset4 : [(.M_1 : obj), (.N_1 : obj), (subsetev2_1
       : [(x_2 : obj), (memberev_2 : that
          x_2 E .M_1) => (--- : that x_2
          E .N_1)]) => (--- : that .M_1
       << .N_1)]


   {move 0}
end Lestrade execution
```