

Implementation of Zermelo's work of 1908 in Lestrade: Part III, opening of Zermelo well-ordering theorem argument

M. Randall Holmes

March 14, 2020

1 Introduction

This document was originally titled as an essay on the proposition that mathematics is what can be done in Automath (as opposed to what can be done in ZFC, for example). Such an essay is still in my mind, but this particular document has transformed itself into the large project of implementing Zermelo's two important set theory papers of 1908 in Lestrade, with the further purpose of exploring the actual capabilities of Zermelo's system of 1908 as a mathematical foundation, which we think are perhaps underrated.

This is a new version of this document in modules, designed to make it possible to work more efficiently without repeated execution of slow log files when they do not need to be revisited.

The new version in March 2020 is designed for the new release of Lestrade.

2 Zermelo's 1908 proof of the well-ordering theorem

I am now going to carry out or at least attempt a significant piece of mathematics in Lestrade. I shall attempt to directly translate Zermelo's 1908 proof of the well-ordering theorem (which was published at about the same time as the axiomatization implemented above: they are intimately connected) into a Lestrade proof.

Zermelo starts by stating prerequisites which are found in the axiomatization. Point I: he requires the axiom of Separation, stated above. He points out as an important corollary the existence of relative complements.

As point II he requires the existence of power sets, provided by us in the axiomatization above.

As point III, he notes that Separation implies the existence of intersections of (nonempty) sets.

In earlier versions, declarations of the above points appeared here, but we have moved them back into the second file, which implements the axiomatics paper.

Zermelo states the following Theorem, the central result in his argument (the well ordering theorem follows immediately from this theorem and the axiom of choice, as we will verify below).

Theorem: If with every nonempty subset of a set M an element of the subset is associated by some law as “distinguished element”, then $\mathcal{P}(x)$, the set of all subsets of M , possesses one and only one subset \mathbf{M} such that to every arbitrary subset P of M there always corresponds one and only one element P_0 of M that includes P as a subset and contains an element of P as its distinguished element. The set M is well-ordered by \mathbf{M} .

The apparent second-order quality of this theorem is dispelled by the proof in Lestrade, in which we see how we can introduce the “law” referred to as a hypothetical without in fact allowing quantification over such laws (and nonetheless successfully carry out the proof of the corollary).

We declare the hypotheses of the theorem, the set M and the unspecified law that, given a subset S of M , allows us to select a distinguished element of S .

begin Lestrade execution

>>> comment load whatismath2

{move 1}

```

>>> clearcurrent

{move 1}

>>> declare M obj

M : obj

{move 1}

>>> declare Misset that Isset M

Misset : that Isset (M)

{move 1}

>>> open

{move 2}

>>> declare S obj

S : obj

{move 2}

>>> declare x obj

x : obj

```

```

{move 2}

>>> declare subsetev that S <= M

subsetev : that S <= M

{move 2}

>>> declare ineq that Exists [x => \
    x E S]

ineq : that Exists [(x_2 : obj) =>
    ({def} x_2 E S : prop)]

{move 2}

>>> postulate thelaw S : obj

thelaw : [(S_1 : obj) => (--- : obj)]

{move 1}

>>> postulate thelawchooses subsetev \
    ineq : that (thelaw S) E S

thelawchooses : [(S_1 : obj), (subsetev_1
    : that S_1 <= M), (ineq_1 : that
    Exists [(x_3 : obj) =>
        ({def} x_3 E S_1 : prop)]) =>
    (--- : that thelaw (S_1) E S_1)]

```

```
{move 1}
```

```
>>> close
```

```
{move 1}
```

```
end Lestrade execution
```

It appears for direct implementation of Zermelo's argument that the choice of a distinguished element should return something arbitrary when the argument is empty, and further it is convenient for the choice to work for any set (or indeed atom) at all, formally, though we make no assumptions about the result. Otherwise we need axioms handling proof indifference and there are other embarrassments.

This seems to be a general fact: operations taking sets to sets (or more generally, sets and atoms to sets and atoms) should be universal, or we end up stumbling over the Lestrade type system.

```
begin Lestrade execution
```

```
>>> open
```

```
{move 2}
```

```
>>> declare S obj
```

```
S : obj
```

```
{move 2}
```

```
>>> declare subsetev that S <= M
```

```

subselev : that  $S \leq M$ 

{move 2}

>>> define prime1 S : Complement (S, Usc \
    (thelaw S))

prime1 : [(S_1 : obj) =>
    ({def} S_1 Complement Usc (thelaw
    (S_1)) : obj)]

prime1 : [(S_1 : obj) => (--- : obj)]

{move 1}

>>> save

{move 2}

>>> close

{move 1}

>>> declare S1 obj

S1 : obj

{move 1}

```

```

>>> define prime2 thelaw, S1 : prime1 \
      S1

prime2 : [(thelaw_1 : [(S_2 : obj) =>
      (--- : obj)]), (S1_3 : obj) =>
      ({def} S1_3 Complement Usc (thelaw_1
      (S1_3)) : obj)]

prime2 : [(thelaw_1 : [(S_2 : obj) =>
      (--- : obj)]), (S1_3 : obj) =>
      (--- : obj)]

{move 0}

>>> open

      {move 2}

>>> define prime S : prime2 thelaw, S

prime : [(S_1 : obj) =>
      ({def} prime2 (thelaw, S_1) : obj)]

prime : [(S_1 : obj) => (--- : obj)]

      {move 1}
end Lestrade execution

```

An important operation in Zermelo's argument takes each subset A of M to the subset $A' = A \setminus \{a\}$, where a is the distinguished element of A if A is nonempty. This is defined above. Below, we prove its natural comprehension

axiom.

begin Lestrade execution

>>> open

{move 3}

>>> declare u obj

u : obj

{move 3}

>>> open

{move 4}

>>> declare hyp1 that $u \in \text{prime} \setminus S$

hyp1 : that $u \in \text{prime} (S)$

{move 4}

>>> declare hyp2 that $(u \in S) \ \& \ \sim (u = \text{thelaw} \setminus S)$

hyp2 : that $(u \in S) \ \& \ \sim (u = \text{thelaw} (S))$


```
{move 4}
```

```
>>> define line1 hyp1 : Iff1 \
      (hyp1, Ui (u, Compax (S, Usc \
      (thelaw S))))
```

```
line1 : [(hyp1_1 : that u E prime
      (S)) =>
      ({def} hyp1_1 Iff1 u Ui S Compax
      Usc (thelaw (S)) : that
      (u E S) & ~ (u E Usc (thelaw
      (S))))]
```

```
line1 : [(hyp1_1 : that u E prime
      (S)) => (--- : that (u E S) & ~ (u E Usc
      (thelaw (S))))]
```

```
{move 3}
```

```
>>> define line2 hyp1 : Simp2 \
      (line1 hyp1)
```

```
line2 : [(hyp1_1 : that u E prime
      (S)) =>
      ({def} Simp2 (line1 (hyp1_1)) : that
      ~ (u E Usc (thelaw (S))))]
```

```
line2 : [(hyp1_1 : that u E prime
      (S)) => (--- : that ~ (u E Usc
      (thelaw (S))))]
```

```
{move 3}
```

```
>>> open
```

```
{move 5}
```

```
>>> declare hyp3 that u = thelaw \  
S
```

```
hyp3 : that u = thelaw (S)
```

```
{move 5}
```

```
>>> define line3 : Inusc2 \  
(thelaw S)
```

```
line3 : [  
  ({def} Inusc2 (thelaw  
    (S)) : that thelaw (S) E thelaw  
    (S) ; thelaw (S))]
```

```
line3 : that thelaw (S) E thelaw  
(S) ; thelaw (S)
```

```
{move 4}
```

```
>>> declare v obj
```

```
v : obj
```

```
{move 5}
```

```
>>> define line4 hyp3 : Subs \
      (Eqsymm hyp3, [v => v E (Usc \
        (thelaw S))], line3)
```

```
line4 : [(hyp3_1 : that
  u = thelaw (S)) =>
  ({def} Subs (Eqsymm (hyp3_1), [(v_2
    : obj) =>
    ({def} v_2 E Usc (thelaw
      (S)) : prop)], line3) : that
  u E Usc (thelaw (S)))]
```

```
line4 : [(hyp3_1 : that
  u = thelaw (S)) => (---
  : that u E Usc (thelaw
    (S)))]
```

```
{move 4}
```

```
>>> define line5 hyp3 : line4 \
      hyp3 Mp line2 hyp1
```

```
line5 : [(hyp3_1 : that
  u = thelaw (S)) =>
  ({def} line4 (hyp3_1) Mp
  line2 (hyp1) : that ??)]
```

```
line5 : [(hyp3_1 : that
  u = thelaw (S)) => (---
  : that ??)]
```

```

{move 4}

>>> close

{move 4}

>>> define line6 hyp1 : Conj \
      (Simp1 line1 hyp1, Negintro \
      line5)

line6 : [(hyp1_1 : that u E prime
      (S)) =>
      ({def} Simp1 (line1 (hyp1_1)) Conj
      Negintro ([hyp3_5 : that
      u = thelaw (S)) =>
      ({def} Subs (Eqsymm (hyp3_5), [(v_7
      : obj) =>
      ({def} v_7 E Usc (thelaw
      (S)) : prop)], Inusc2
      (thelaw (S))) Mp line2
      (hyp1_1) : that ??)]) : that
      (u E S) & ~ (u = thelaw
      (S)))]

line6 : [(hyp1_1 : that u E prime
      (S)) => (--- : that (u E S) & ~ (u = thelaw
      (S)))]

{move 3}

>>> open

```

```

{move 5}

>>> declare hyp4 that u E Usc \
      (thelaw S)

hyp4 : that u E Usc (thelaw
      (S))

{move 5}

>>> define line8 hyp4 : Mp \
      (Inusc1 hyp4, Simp2 hyp2)

line8 : [(hyp4_1 : that
          u E Usc (thelaw (S))) =>
          ({def} Inusc1 (hyp4_1) Mp
           Simp2 (hyp2) : that ??)]

line8 : [(hyp4_1 : that
          u E Usc (thelaw (S))) =>
          (--- : that ??)]

{move 4}

>>> close

{move 4}

>>> define line9 hyp2 : Conj \
      (Simp1 hyp2, Negintro line8)

```

```

line9 : [(hyp2_1 : that (u E S) & ~ (u = thelaw
(S))) =>
({def} Simp1 (hyp2_1) Conj
Negintro [(hyp4_2 : that
u E Usc (thelaw (S))) =>
({def} Inusc1 (hyp4_2) Mp
Simp2 (hyp2_1) : that
??)] : that (u E S) & ~ (u E Usc
(thelaw (S)))))]

```

```

line9 : [(hyp2_1 : that (u E S) & ~ (u = thelaw
(S))) => (--- : that
(u E S) & ~ (u E Usc (thelaw
(S)))))]

```

```

{move 3}

```

```

>>> define line10 hyp2 : Iff2 \
(line9 hyp2, Ui (u, Compax \
(S, Usc (thelaw S))))

```

```

line10 : [(hyp2_1 : that (u E S) & ~ (u = thelaw
(S))) =>
({def} line9 (hyp2_1) Iff2
u Ui S Compax Usc (thelaw
(S)) : that u E S Complement
Usc (thelaw (S)))]

```

```

line10 : [(hyp2_1 : that (u E S) & ~ (u = thelaw
(S))) => (--- : that
u E S Complement Usc (thelaw
(S)))]

```

```

{move 3}

>>> close

{move 3}

>>> define bothways u : Dediff line6, line10

bothways : [(u_1 : obj) =>
  ({def} Dediff ([hyp1_6 : that
    u_1 E prime (S)) =>
    ({def} Simp1 (hyp1_6 Iff1
      u_1 Ui S Compax Usc (thelaw
        (S))) Conj Negintro ([hyp3_8
          : that u_1 = thelaw (S)) =>
          ({def} Subs (Eqsymm (hyp3_8), [(v_10
            : obj) =>
              ({def} v_10 E Usc (thelaw
                (S)) : prop)], Inusc2
                (thelaw (S))) Mp Simp2
                (hyp1_6 Iff1 u_1 Ui S Compax
                  Usc (thelaw (S))) : that
                  ??)]) : that (u_1 E S) & ~ (u_1
                    = thelaw (S))], [(hyp2_6
                      : that (u_1 E S) & ~ (u_1
                        = thelaw (S))) =>
                      ({def} Simp1 (hyp2_6) Conj
                        Negintro ([hyp4_9 : that
                          u_1 E Usc (thelaw (S))) =>
                          ({def} Inusc1 (hyp4_9) Mp
                            Simp2 (hyp2_6) : that
                            ??)]) Iff2 u_1 Ui S Compax
                          Usc (thelaw (S)) : that
                          u_1 E S Complement Usc (thelaw
                            (S)))] : that (u_1

```

```

      E prime (S)) == (u_1 E S) & ~ (u_1
      = thelaw (S)))]

bothways : [(u_1 : obj) => (---
      : that (u_1 E prime (S)) ==
      (u_1 E S) & ~ (u_1 = thelaw
      (S)))]

{move 2}

>>> close

{move 2}

>>> define primeax subsetev : Ug bothways

primeax : [(S_1 : obj), (subsetev_1
      : that S_1 <= M) =>
      ({def} Ug [(u_2 : obj) =>
      ({def} Dediff [(hyp1_3 : that
      u_2 E prime (S_1)) =>
      ({def} Simp1 (hyp1_3 Iff1
      u_2 Ui S_1 Compax Usc (thelaw
      (S_1))) Conj Negintro
      [(hyp3_5 : that u_2 = thelaw
      (S_1)) =>
      ({def} Subs (Eqsymm (hyp3_5), [(v_7
      : obj) =>
      ({def} v_7 E Usc (thelaw
      (S_1)) : prop)], Inusc2
      (thelaw (S_1))) Mp
      Simp2 (hyp1_3 Iff1 u_2
      Ui S_1 Compax Usc (thelaw
      (S_1))) : that ??)]) : that

```



```

(u_2 E .S_1) & ~ (u_2 = thelaw
(.S_1)))], [(hyp2_3
: that (u_2 E .S_1) & ~ (u_2
= thelaw (.S_1))) =>
({def} Simp1 (hyp2_3) Conj
Negintro ([hyp4_6 : that
u_2 E Usc (thelaw (.S_1))) =>
({def} Inusc1 (hyp4_6) Mp
Simp2 (hyp2_3) : that
??)]) Iff2 u_2 Ui .S_1
Compax Usc (thelaw (.S_1)) : that
u_2 E .S_1 Complement Usc
(thelaw (.S_1)))] : that
(u_2 E prime (.S_1)) == (u_2
E .S_1) & ~ (u_2 = thelaw (.S_1)))] : that
Forall ([x'_11 : obj) =>
({def} (x'_11 E prime (.S_1)) ==
(x'_11 E .S_1) & ~ (x'_11
= thelaw (.S_1)) : prop)]))]]

```

```

primeax : [(S_1 : obj), (subsevev_1
: that .S_1 <= M) => (--- : that
Forall ([x'_11 : obj) =>
({def} (x'_11 E prime (.S_1)) ==
(x'_11 E .S_1) & ~ (x'_11
= thelaw (.S_1)) : prop)]))]

```

```
{move 1}
```

```
>>> close
```

```
{move 1}
```

```
end Lestrade execution
```

Now we are ready to define the central concept of this central argument,

the idea of a Θ -chain. The definition of a Θ -chain has four clauses, and there are a tiresome number of occurrences in this document of proofs of the four statements required to show that a particular set is a Θ -chain.

begin Lestrade execution

```
>>> declare C11 obj
```

```
C11 : obj
```

```
{move 1}
```

```
>>> declare D11 obj
```

```
D11 : obj
```

```
{move 1}
```

```
>>> declare F11 obj
```

```
F11 : obj
```

```
{move 1}
```

```
>>> define thetachain1 M thelaw, C11 \
      : (M E C11) & (C11 <= Sc (M)) & Forall \
      [D11 => (D11 E C11) -> (prime D11) E C11] & Forall \
      [D11 => Forall [F11 => ((D11 <= C11) & (F11 \
        E D11)) -> (Intersection D11 \
        F11) E C11]]
```

```

thetachain1 : [(M_1 : obj), (thelaw_1
  : [(S_2 : obj) => (--- : obj)]), (C11_3
  : obj) =>
  ({def} (M_1 E C11_3) & (C11_3 <=<=
  Sc (M_1)) & Forall ([(D11_7 : obj) =>
    ({def} (D11_7 E C11_3) -> prime2
    (thelaw_1, D11_7) E C11_3 : prop)]) & Forall
    ([(D11_7 : obj) =>
      ({def} Forall ([(F11_8 : obj) =>
        ({def} ((D11_7 <=<= C11_3) & F11_8
        E D11_7) -> (D11_7 Intersection
        F11_8) E C11_3 : prop)]) : prop)]) : prop)]

```

```

thetachain1 : [(M_1 : obj), (thelaw_1
  : [(S_2 : obj) => (--- : obj)]), (C11_3
  : obj) => (--- : prop)]

```

```
{move 0}
```

```
>>> open
```

```
{move 2}
```

```
>>> declare C obj
```

```
C : obj
```

```
{move 2}
```

```
>>> declare D obj
```

```
D : obj
```

```
{move 2}
```

```
>>> declare F obj
```

```
F : obj
```

```
{move 2}
```

```
>>> define thetachain C : thetachain1 \
      M thelaw, C
```

```
thetachain : [(C_1 : obj) =>
      ({def} thetachain1 (M, thelaw, C_1) : prop)]
```

```
thetachain : [(C_1 : obj) => (---
      : prop)]
```

```
{move 1}
```

```
>>> declare thetaev that thetachain \
      C
```

```
thetaev : that thetachain (C)
```

```
{move 2}
```

```
>>> declare G obj
```

```

G : obj

{move 2}

>>> declare ginev that G E C

ginev : that G E C

{move 2}

>>> define Setsinchains1 thetaev ginev \
      : Simp1 (Simp2 (Iff1 (Mp (ginev, Ui \
      (G, Simp1 (Simp1 (Simp2 thetaev))))), Ui \
      G Scthm M)))

Setsinchains1 : [(C_1 : obj), (thetaev_1
      : that thetachain (C_1)), (G_1
      : obj), (ginev_1 : that G_1
      E C_1) =>
      ({def} Simp1 (Simp2 (ginev_1
      Mp G_1 Ui Simp1 (Simp1 (Simp2
      (thetaev_1))) Iff1 G_1 Ui Scthm
      (M))) : that Isset (G_1)))]

Setsinchains1 : [(C_1 : obj), (thetaev_1
      : that thetachain (C_1)), (G_1
      : obj), (ginev_1 : that G_1
      E C_1) => (--- : that Isset (G_1)))]

{move 1}

```

```

>>> save

{move 2}

>>> close

{move 1}

>>> declare C10 obj

C10 : obj

{move 1}

>>> declare thetaev10 that thetachain \
      C10

thetaev10 : that thetachain (C10)

{move 1}

>>> declare G10 obj

G10 : obj

{move 1}

>>> declare ginev10 that G10 E C10

```

```
ginev10 : that G10 E C10
```

```
{move 1}
```

```
>>> define Setsinchains2 Misset thelawchooses, thetaev10 \
      ginev10 : Setsinchains1 thetaev10 ginev10
```

```
Setsinchains2 : [(M_1 : obj), (Misset_1
  : that Isset (M_1)), (.thelaw_1
  : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
  : [(S_2 : obj), (subsestev_2 : that
    .S_2 <=<= .M_1), (inev_2 : that
    Exists [(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])]) =>
  (--- : that .thelaw_1 (.S_2) E .S_2)], (.C10_3
  : obj), (thetaev10_3 : that thetachain1
  (.M_1, .thelaw_1, .C10_3)), (.G10_3
  : obj), (ginev10_3 : that .G10_3
  E .C10_3) =>
  ({def} Simp1 (Simp2 (ginev10_3 Mp
  .G10_3 Ui Simp1 (Simp1 (Simp2 (thetaev10_3))) Iff1
  .G10_3 Ui Scthm (.M_1))) : that
  Isset (.G10_3))]
```

```
Setsinchains2 : [(M_1 : obj), (Misset_1
  : that Isset (M_1)), (.thelaw_1
  : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
  : [(S_2 : obj), (subsestev_2 : that
    .S_2 <=<= .M_1), (inev_2 : that
    Exists [(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])]) =>
  (--- : that .thelaw_1 (.S_2) E .S_2)], (.C10_3
  : obj), (thetaev10_3 : that thetachain1
  (.M_1, .thelaw_1, .C10_3)), (.G10_3
  : obj), (ginev10_3 : that .G10_3
```

```

      E .C10_3) => (--- : that Isset (.G10_3))]]

{move 0}

>>> open

{move 2}

>>> define Setsinchains thetaev ginev \
      : Setsinchains2 Misset, thelawchooses, thetaev \
      ginev

Setsinchains : [(C_1 : obj), (thetaev_1
      : that thetachain (C_1)), (G_1
      : obj), (ginev_1 : that G_1
      E .C_1) =>
      ({def} Setsinchains2 (Misset, thelawchooses, thetaev_1, ginev_1) : th
      Isset (G_1))]]

Setsinchains : [(C_1 : obj), (thetaev_1
      : that thetachain (C_1)), (G_1
      : obj), (ginev_1 : that G_1
      E .C_1) => (--- : that Isset (G_1))]]

{move 1}

>>> close

{move 1}
end Lestrade execution

```

We did some extra work to ensure that `thetachain` is defined in terms

of a notion at move 0 which will not be expanded everywhere it occurs.

We then proved that each element of a Θ -chain is a set and again did work to control definitional expansion.

We then need to prove that $\mathcal{P}(M)$ is a Θ -chain.

```
begin Lestrade execution
```

```
>>> open
```

```
{move 2}
```

```
>>> open
```

```
{move 3}
```

```
>>> define line1 : Misset Mp Ui \
      M Inownpowerset
```

```
line1 : Misset Mp M Ui Inownpowerset
```

```
line1 : that M E Sc (M)
```

```
{move 2}
```

```
>>> define line2 : (Misset Mp Ui \
      M Scofsetisset) Mp Ui Sc M Subsetrefl
```

```
line2 : Misset Mp M Ui Scofsetisset
      Mp Sc (M) Ui Subsetrefl
```

```
line2 : that Sc (M) <=< Sc (M)
```

```
{move 2}
end Lestrade execution
```

The first two lines establish the first two of the four points needed to show that $\mathcal{P}(M)$ is a Θ -chain.

```
begin Lestrade execution
```

```
>>> open
```

```
{move 4}
```

```
>>> declare u obj
```

```
u : obj
```

```
{move 4}
```

```
>>> open
```

```
{move 5}
```

```
>>> declare usinev that u E Sc \
      M
```

```
usinev : that u E Sc (M)
```

```
{move 5}
```

```

>>> define line3 : Fixform \
      (Isset (prime u), Separation3 \
      (Refleq prime u))

line3 : [
  ({def} Isset (prime (u)) Fixform
  Separation3 (Refleq (prime
  (u))) : that Isset
  (prime (u))))]

line3 : that Isset (prime
(u))

{move 4}
end Lestrade execution

```

Note the devious use of **Separation3** to avoid having to write out the defining predicate of the prime operation. The use of the axiom **Separation2** is essential: the result of applying the prime operation might be empty, and we do want it to be a set.

```

begin Lestrade execution

>>> define line4 usinev : Simp1 \
      (Simp2 (Iff1 (usinev, Ui \
      u (Scthm M))))

line4 : [(usinev_1 : that
  u E Sc (M)) =>
  ({def} Simp1 (Simp2 (usinev_1
  Iff1 u Ui Scthm (M))) : that
  Isset (u))]

```

```

line4 : [(usinev_1 : that
          u E Sc (M)) => (---
          : that Isset (u))]

```

```

{move 4}

```

```

>>> open

```

```

{move 6}

```

```

>>> declare v obj

```

```

v : obj

```

```

{move 6}

```

```

>>> open

```

```

{move 7}

```

```

>>> declare vinev that \
      v E prime u

```

```

vinev : that v E prime
      (u)

```

```

{move 7}

```

```

>>> define line5 : Ui \

```

```

v primeax (Iff1 (usinev, Ui \
u Scthm M))

```

```

line5 : v Ui primeax
(usinev Iff1 u Ui Scthm
(M))

```

```

line5 : that (v E prime
(u)) == (v E u) & ~ (v = thelaw
(u))

```

```

{move 6}

```

```

>>> define line6 vinev \
: Simp1 (Iff1 (vinev, line5))

```

```

line6 : [(vinev_1
: that v E prime
(u)) =>
({def} Simp1 (vinev_1
Iff1 line5) : that
v E u)]

```

```

line6 : [(vinev_1
: that v E prime
(u)) => (---
: that v E u)]

```

```

{move 6}

```

```

>>> define line7 vinev \
: Mp line6 vinev (Ui \

```

```

v Simp1 (Iff1 (usinev, Ui \
u Scthm M)))

```

```

line7 : [(vinev_1
: that v E prime
(u)) =>
({def} line6 (vinev_1) Mp
v Ui Simp1 (usinev
Iff1 u Ui Scthm (M)) : that
v E M)]

```

```

line7 : [(vinev_1
: that v E prime
(u)) => (---
: that v E M)]

```

```

{move 6}

```

```

>>> close

```

```

{move 6}

```

```

>>> define line8 v : Ded \
line7

```

```

line8 : [(v_1 : obj) =>
({def} Ded ([vinev_8
: that v_1 E prime
(u)) =>
({def} Simp1 (vinev_8
Iff1 v_1 Ui primeax
(usinev Iff1 u Ui
Scthm (M)))) Mp

```

```

      v_1 Ui Simp1 (usinev
      Iff1 u Ui Scthm (M)) : that
      v_1 E M])) : that
      (v_1 E prime (u)) ->
      v_1 E M)]

```

```

line8 : [(v_1 : obj) =>
  (--- : that (v_1 E prime
  (u)) -> v_1 E M)]

```

```

{move 5}

```

```

>>> close

```

```

{move 5}

```

```

>>> define line9 usinev : Ug \
  line8

```

```

line9 : [(usinev_1 : that
  u E Sc (M)) =>
  ({def} Ug ([v_2 : obj) =>
    ({def} Ded ([vinev_3
      : that v_2 E prime
      (u)) =>
      ({def} Simp1 (vinev_3
        Iff1 v_2 Ui primeax
        (usinev_1 Iff1 u Ui
        Scthm (M))) Mp
        v_2 Ui Simp1 (usinev_1
        Iff1 u Ui Scthm (M)) : that
        v_2 E M])) : that
      (v_2 E prime (u)) ->
      v_2 E M])) : that

```

```

Forall ([ (x'_12 : obj) =>
  ({def} (x'_12 E prime
    (u)) -> x'_12 E M : prop) ]))

```

```

line9 : [(usinev_1 : that
  u E Sc (M)) => (---
  : that Forall ([ (x'_12
    : obj) =>
    ({def} (x'_12 E prime
      (u)) -> x'_12 E M : prop) ]))]

```

```

{move 4}

```

```

>>> define line10 usinev : Fixform \
  ((prime u) <=& M, Conj \
  (line9 usinev, Conj (line3, Misset)))

```

```

line10 : [(usinev_1 : that
  u E Sc (M)) =>
  ({def} (prime (u) <=&
  M) Fixform line9 (usinev_1) Conj
  line3 Conj Misset : that
  prime (u) <=& M)]

```

```

line10 : [(usinev_1 : that
  u E Sc (M)) => (---
  : that prime (u) <=&
  M)]

```

```

{move 4}

```

```

>>> define line11 usinev : Iff2 \
  (line10 usinev, Ui (prime \

```



```
u, Scthm M))
```

```
line11 : [(usinev_1 : that
  u E Sc (M)) =>
  ({def} line10 (usinev_1) Iff2
  prime (u) Ui Scthm (M) : that
  prime (u) E Sc (M))]
```

```
line11 : [(usinev_1 : that
  u E Sc (M)) => (---
  : that prime (u) E Sc
  (M))]
```

```
{move 4}
```

```
>>> close
```

```
{move 4}
```

```
>>> define line12 u : Ded line11
```

```
line12 : [(u_1 : obj) =>
  ({def} Ded ([usinev_6
  : that u_1 E Sc (M)) =>
  ({def} ((prime (u_1) <=<=
  M) Fixform Ug ([v_10
  : obj) =>
  ({def} Ded ([vinev_11
  : that v_10 E prime
  (u_1)) =>
  ({def} Simp1 (vinev_11
  Iff1 v_10 Ui primeax
  (usinev_6 Iff1 u_1
```

```

      Ui Scthm (M))) Mp
      v_10 Ui Simp1 (usinev_6
      Iff1 u_1 Ui Scthm
      (M)) : that v_10
      E M]]) : that
      (v_10 E prime (u_1)) ->
      v_10 E M]]) Conj
      (Isset (prime (u_1)) Fixform
      Separation3 (Refleq (prime
      (u_1)))) Conj Misset) Iff2
      prime (u_1) Ui Scthm
      (M) : that prime (u_1) E Sc
      (M))]]) : that (u_1
      E Sc (M)) -> prime (u_1) E Sc
      (M))]]

```

```

line12 : [(u_1 : obj) => (---
      : that (u_1 E Sc (M)) ->
      prime (u_1) E Sc (M))]]

```

```

{move 3}

```

```

>>> close

```

```

{move 3}

```

```

>>> define line13 : Ug line12

```

```

line13 : Ug ([ (u_2 : obj) =>
      ({def} Ded ([ (usinev_3 : that
      u_2 E Sc (M)) =>
      ({def} ((prime (u_2) <=<=
      M) Fixform Ug ([ (v_7 : obj) =>
      ({def} Ded ([ (vinev_8

```

```

      : that v_7 E prime (u_2)) =>
      ({def} Simp1 (vinev_8
      Iff1 v_7 Ui primeax
      (usinev_3 Iff1 u_2
      Ui Scthm (M))) Mp
      v_7 Ui Simp1 (usinev_3
      Iff1 u_2 Ui Scthm (M)) : that
      v_7 E M]]) : that
      (v_7 E prime (u_2)) ->
      v_7 E M]]) Conj (Isset
      (prime (u_2)) Fixform
      Separation3 (Refleq (prime
      (u_2)))) Conj Misset) Iff2
      prime (u_2) Ui Scthm (M) : that
      prime (u_2) E Sc (M))]]) : that
      (u_2 E Sc (M)) -> prime (u_2) E Sc
      (M))]])

```

```

line13 : that Forall ([ (x'_17
      : obj) =>
      ({def} (x'_17 E Sc (M)) ->
      prime (x'_17) E Sc (M) : prop)])

```

```

      {move 2}
end Lestrade execution

```

Here is the third statement needed to verify that $\mathcal{P}(M)$ is a Θ -chain.

```

begin Lestrade execution

```

```

>>> open

```

```

      {move 4}

```

```

>>> declare u obj

u : obj

{move 4}

>>> open

{move 5}

>>> declare v obj

v : obj

{move 5}

>>> open

{move 6}

>>> declare hyp that (u <= \
    Sc M) & v E u

hyp : that (u <= Sc (M)) & v E u

{move 6}

>>> define line14 hyp : Simp2 \
    hyp Mp Intax u v

```

```

line14 : [(hyp_1 : that
  (u <=< Sc (M)) & v E u) =>
  ({def} Simp2 (hyp_1) Mp
  u Intax v : that Forall
  [(x''_2 : obj) =>
    ({def} (x''_2 E u Intersection
    v) == Forall [(B1_4
      : obj) =>
      ({def} (B1_4
      E u) -> x''_2
      E B1_4 : prop))] : prop))]]

```

```

line14 : [(hyp_1 : that
  (u <=< Sc (M)) & v E u) =>
  (--- : that Forall
  [(x''_2 : obj) =>
    ({def} (x''_2 E u Intersection
    v) == Forall [(B1_4
      : obj) =>
      ({def} (B1_4
      E u) -> x''_2
      E B1_4 : prop))] : prop))]]

```

```
{move 5}
```

```
>>> open
```

```
{move 7}
```

```
>>> declare w obj
```

```
w : obj
```

```
{move 7}
```

```
>>> open
```

```
{move 8}
```

```
>>> declare hyp2 \  
      that w E Intersection \  
      u v
```

```
hyp2 : that w E u Intersection  
v
```

```
{move 8}
```

```
>>> define line15 \  
      hyp2 : Mp (Simp2 \  
      hyp, Ui v (Iff1 \  
      (hyp2, Ui w line14 \  
      hyp)))
```

```
line15 : [(hyp2_1  
      : that w E u Intersection  
      v) =>  
      ({def} Simp2  
      (hyp) Mp v Ui  
      hyp2_1 Iff1 w Ui  
      line14 (hyp) : that  
      w E v)]
```

```
line15 : [(hyp2_1  
      : that w E u Intersection
```

```

v) => (--- : that
w E v)]

```

```

{move 7}

```

```

>>> define line16 \
      : Mp (Simp2 hyp, Ui \
      v Simp1 (Simp1 hyp))

```

```

line16 : Simp2 (hyp) Mp
v Ui Simp1 (Simp1
(hyp))

```

```

line16 : that v E Sc
(M)

```

```

{move 7}

```

```

>>> define line17 \
      : Iff1 (line16, Ui \
      v Scthm M)

```

```

line17 : [
  ({def} line16
  Iff1 v Ui Scthm
  (M) : that v <=<=
  M)]

```

```

line17 : that v <=<=
M

```

```
{move 7}
```

```
>>> define line18 \  
      hyp2 : Mp (line15 \  
      hyp2, Ui w Simp1 \  
      line17)
```

```
line18 : [(hyp2_1  
      : that w E u Intersection  
      v) =>  
      ({def} line15  
      (hyp2_1) Mp  
      w Ui Simp1 (line17) : that  
      w E M)]
```

```
line18 : [(hyp2_1  
      : that w E u Intersection  
      v) => (--- : that  
      w E M)]
```

```
{move 7}
```

```
>>> close
```

```
{move 7}
```

```
>>> define line19 w : Ded \  
      line18
```

```
line19 : [(w_1 : obj) =>  
      ({def} Ded ([hyp2_10  
      : that w_1 E u Intersection  
      v) =>
```



```

({def} Simp2
(hyp) Mp v Ui
hyp2_10 Iff1 w_1
Ui line14 (hyp) Mp
w_1 Ui Simp1 (Simp2
(hyp) Mp v Ui
Simp1 (Simp1
(hyp)) Iff1
v Ui Scthm (M)) : that
w_1 E M]]) : that
(w_1 E u Intersection
v) -> w_1 E M)]

```

```

line19 : [(w_1 : obj) =>
  (--- : that (w_1
  E u Intersection
  v) -> w_1 E M)]

```

```

{move 6}

```

```

>>> close

```

```

{move 6}

```

```

>>> define line20 hyp : Ug \
  line19

```

```

line20 : [(hyp_1 : that
  (u <=< Sc (M)) & v E u) =>
  ({def} Ug ([w_2
  : obj) =>
  ({def} Ded ([hyp2_3
  : that w_2 E u Intersection
  v) =>

```

```

      ({def} Simp2
      (hyp_1) Mp v Ui
      hyp2_3 Iff1 w_2
      Ui line14 (hyp_1) Mp
      w_2 Ui Simp1 (Simp2
      (hyp_1) Mp v Ui
      Simp1 (Simp1
      (hyp_1)) Iff1
      v Ui Scthm (M)) : that
      w_2 E M])) : that
      (w_2 E u Intersection
      v) -> w_2 E M])) : that
      Forall ([ (x'_13 : obj) =>
      ({def} (x'_13 E u Intersection
      v) -> x'_13 E M : prop))]))]

```

```

line20 : [(hyp_1 : that
      (u <= Sc (M)) & v E u) =>
      (--- : that Forall
      [(x'_13 : obj) =>
      ({def} (x'_13 E u Intersection
      v) -> x'_13 E M : prop))]])]

```

```

{move 5}

```

```

>>> define line21 : Fixform \
      (Isset (Intersection \
      u v), Separation3 (Refleq \
      (Intersection u v)))

```

```

line21 : [
      ({def} Isset (u Intersection
      v) Fixform Separation3
      (Refleq (u Intersection
      v)) : that Isset (u Intersection

```

```
v))]
```

```
line21 : that Isset (u Intersection
v)
```

```
{move 5}
```

```
>>> define line22 hyp : Fixform \
      ((Intersection u v) <=& \
      M, Conj (line20 hyp, Conj \
      (line21, Misset)))
```

```
line22 : [(hyp_1 : that
      (u <=& Sc (M)) & v E u) =>
      ({def} ((u Intersection
      v) <=& M) Fixform
      line20 (hyp_1) Conj
      line21 Conj Misset : that
      (u Intersection v) <=&
      M)]
```

```
line22 : [(hyp_1 : that
      (u <=& Sc (M)) & v E u) =>
      (--- : that (u Intersection
      v) <=& M)]
```

```
{move 5}
```

```
>>> define line23 hyp : Iff2 \
      (line22 hyp, Ui (Intersection \
      u v, Scthm M))
```

```

line23 : [(hyp_1 : that
  (u <=<= Sc (M)) & v E u) =>
  ({def} line22 (hyp_1) Iff2
  (u Intersection v) Ui
  Scthm (M) : that (u Intersection
  v) E Sc (M)))]

```

```

line23 : [(hyp_1 : that
  (u <=<= Sc (M)) & v E u) =>
  (--- : that (u Intersection
  v) E Sc (M)))]

```

```

{move 5}

```

```

>>> close

```

```

{move 5}

```

```

>>> define line24 v : Ded \
  line23

```

```

line24 : [(v_1 : obj) =>
  ({def} Ded ([hyp_8
    : that (u <=<= Sc (M)) & v_1
    E u) =>
    ({def} (((u Intersection
    v_1) <=<= M) Fixform
    Ug ([w_12 : obj) =>
      ({def} Ded ([hyp2_13
        : that w_12 E u Intersection
        v_1) =>
        ({def} Simp2
        (hyp_8) Mp v_1
        Ui hyp2_13 Iff1

```

```

w_12 Ui Simp2
(hyp_8) Mp u Intax
v_1 Mp w_12 Ui
Simp1 (Simp2
(hyp_8) Mp v_1
Ui Simp1 (Simp1
(hyp_8)) Iff1
v_1 Ui Scthm (M)) : that
w_12 E M]]) : that
(w_12 E u Intersection
v_1) -> w_12 E M]]) Conj
(Isset (u Intersection
v_1) Fixform Separation3
(Refleq (u Intersection
v_1))) Conj Misset) Iff2
(u Intersection v_1) Ui
Scthm (M) : that (u Intersection
v_1) E Sc (M))]] : that
((u <=< Sc (M)) & v_1
E u) -> (u Intersection
v_1) E Sc (M))]
```

```

line24 : [(v_1 : obj) =>
  (--- : that ((u <=<
  Sc (M)) & v_1 E u) ->
  (u Intersection v_1) E Sc
  (M))]
```

```
{move 4}
```

```
>>> close
```

```
{move 4}
```

```
>>> define line25 u : Ug line24
```

```

line25 : [(u_1 : obj) =>
  ({def} Ug [(v_2 : obj) =>
    ({def} Ded [(hyp_3
      : that (u_1 <=< Sc
        (M)) & v_2 E u_1) =>
      ({def} ((u_1 Intersection
        v_2) <=< M) Fixform
      Ug [(w_7 : obj) =>
        ({def} Ded [(hyp2_8
          : that w_7 E u_1
          Intersection v_2) =>
          ({def} Simp2
            (hyp_3) Mp v_2
            Ui hyp2_8 Iff1
            w_7 Ui Simp2 (hyp_3) Mp
            u_1 Intax v_2
            Mp w_7 Ui Simp1
            (Simp2 (hyp_3) Mp
              v_2 Ui Simp1 (Simp1
                (hyp_3)) Iff1
              v_2 Ui Scthm (M)) : that
              w_7 E M)]) : that
            (w_7 E u_1 Intersection
              v_2) -> w_7 E M)]) Conj
          (Isset (u_1 Intersection
            v_2) Fixform Separation3
            (Refleq (u_1 Intersection
              v_2))) Conj Misset) Iff2
            (u_1 Intersection v_2) Ui
            Scthm (M) : that (u_1
              Intersection v_2) E Sc
              (M)))] : that ((u_1
                <=< Sc (M)) & v_2 E u_1) ->
                (u_1 Intersection v_2) E Sc
                (M)))] : that Forall
                [(x'_19 : obj) =>

```

```

({def} ((u_1 <=< Sc
(M)) & x'_19 E u_1) ->
(u_1 Intersection x'_19) E Sc
(M) : prop)))]

```

```

line25 : [(u_1 : obj) => (---
: that Forall ([(x'_19
: obj) =>
({def} ((u_1 <=< Sc
(M)) & x'_19 E u_1) ->
(u_1 Intersection x'_19) E Sc
(M) : prop)))]

```

```

{move 3}

```

```

>>> close

```

```

{move 3}

```

```

>>> define line26 : Ug line25

```

```

line26 : Ug ([(u_2 : obj) =>
({def} Ug ([(v_3 : obj) =>
({def} Ded ([(hyp_4 : that
(u_2 <=< Sc (M)) & v_3
E u_2) =>
({def} (((u_2 Intersection
v_3) <=< M) Fixform Ug
([(w_8 : obj) =>
({def} Ded ([(hyp2_9
: that w_8 E u_2
Intersection v_3) =>
({def} Simp2 (hyp_4) Mp
v_3 Ui hyp2_9 Iff1

```

```

      w_8 Ui Simp2 (hyp_4) Mp
      u_2 Intax v_3 Mp
      w_8 Ui Simp1 (Simp2
      (hyp_4) Mp v_3
      Ui Simp1 (Simp1
      (hyp_4)) Iff1
      v_3 Ui Scthm (M)) : that
      w_8 E M]]) : that
      (w_8 E u_2 Intersection
      v_3) -> w_8 E M]]) Conj
      (Isset (u_2 Intersection
      v_3) Fixform Separation3
      (Refleq (u_2 Intersection
      v_3))) Conj Misset) Iff2
      (u_2 Intersection v_3) Ui
      Scthm (M) : that (u_2
      Intersection v_3) E Sc
      (M)]] : that ((u_2
      <=< Sc (M)) & v_3 E u_2) ->
      (u_2 Intersection v_3) E Sc
      (M)]] : that Forall
      ([ (x'_3 : obj) =>
      ({def} ((u_2 <=< Sc (M)) & x'_3
      E u_2) -> (u_2 Intersection
      x'_3) E Sc (M) : prop)))]])

line26 : that Forall ([ (x'_20
: obj) =>
({def} Forall ([ (x'_21 : obj) =>
({def} ((x'_20 <=< Sc (M)) & x'_21
E x'_20) -> (x'_20 Intersection
x'_21) E Sc (M) : prop))] : prop)])

{move 2}
end Lestrade execution

```


Here is the fourth and last statement needed to verify that the power set of M is a Θ -chain.

begin Lestrade execution

```
>>> close
```

```
{move 2}
```

```
>>> define thetascm1 : Fixform (thetachain \
  (Sc M), Conj (line1, Conj (line2, Conj \
    (line13, line26))))
```

```
thetascm1 : [
  ({def} thetachain (Sc (M)) Fixform
  Misset Mp M Ui Inownpowerset Conj
  Misset Mp M Ui Scofsetisset Mp Sc
  (M) Ui Subsetrefl Conj Ug ([u_10
    : obj) =>
    ({def} Ded ([usinev_11 : that
      u_10 E Sc (M)) =>
      ({def} ((prime (u_10) <=<=
        M) Fixform Ug ([v_15
          : obj) =>
          ({def} Ded ([vinev_16
            : that v_15 E prime
            (u_10)) =>
            ({def} Simp1 (vinev_16
              Iff1 v_15 Ui primeax
              (usinev_11 Iff1 u_10
                Ui Scthm (M))) Mp
              v_15 Ui Simp1 (usinev_11
                Iff1 u_10 Ui Scthm (M)) : that
                v_15 E M])) : that
              (v_15 E prime (u_10)) ->
```

```

      v_15 E M]]) Conj (Isset
      (prime (u_10)) Fixform
      Separation3 (Refleq (prime
      (u_10)))) Conj Misset) Iff2
      prime (u_10) Ui Scthm (M) : that
      prime (u_10) E Sc (M))]] : that
      (u_10 E Sc (M)) -> prime
      (u_10) E Sc (M))]] Conj
Ug ([ (u_10 : obj) =>
      ({def} Ug ([ (v_11 : obj) =>
      ({def} Ded ([ (hyp_12 : that
      (u_10 <=< Sc (M)) & v_11
      E u_10) =>
      ({def} ((u_10 Intersection
      v_11) <=< M) Fixform
      Ug ([ (w_16 : obj) =>
      ({def} Ded ([ (hyp2_17
      : that w_16 E u_10
      Intersection v_11) =>
      ({def} Simp2 (hyp_12) Mp
      v_11 Ui hyp2_17 Iff1
      w_16 Ui Simp2 (hyp_12) Mp
      u_10 Intax v_11 Mp
      w_16 Ui Simp1 (Simp2
      (hyp_12) Mp v_11
      Ui Simp1 (Simp1
      (hyp_12)) Iff1
      v_11 Ui Scthm (M)) : that
      w_16 E M))]] : that
      (w_16 E u_10 Intersection
      v_11) -> w_16 E M))]] Conj
      (Isset (u_10 Intersection
      v_11) Fixform Separation3
      (Refleq (u_10 Intersection
      v_11))) Conj Misset) Iff2
      (u_10 Intersection v_11) Ui
      Scthm (M) : that (u_10
      Intersection v_11) E Sc

```

```

        (M))] : that ((u_10
        <=< Sc (M)) & v_11 E u_10) ->
        (u_10 Intersection v_11) E Sc
        (M))] : that Forall
    ([ (x'_11 : obj) =>
        ({def} ((u_10 <=< Sc (M)) & x'_11
        E u_10) -> (u_10 Intersection
        x'_11) E Sc (M) : prop)))] : that
    thetachain (Sc (M)))

thetascm1 : that thetachain (Sc (M))

{move 1}

>>> close

{move 1}

>>> define thetascm2 Misset thelawchooses \
    : thetascm1

thetascm2 : [(M_1 : obj), (Misset_1
    : that Isset (.M_1)), (.thelaw_1
    : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
    : [(S_2 : obj), (subsestev_2 : that
        .S_2 <=< .M_1), (inev_2 : that
        Exists ([ (x_4 : obj) =>
            ({def} x_4 E .S_2 : prop)))] =>
        (--- : that .thelaw_1 (.S_2) E .S_2))] =>
    ({def} thetachain1 (.M_1, .thelaw_1, Sc
    (.M_1)) Fixform Misset_1 Mp .M_1
    Ui Inownpowerset Conj Misset_1 Mp .M_1
    Ui Scofsetisset Mp Sc (.M_1) Ui Subsetrefl
    Conj Ug ([ (u_8 : obj) =>

```

```

({def} Ded ([usinev_9 : that
u_8 E Sc (.M_1)) =>
({def} ((prime2 (.thelaw_1, u_8) <=&=
.M_1) Fixform Ug ([v_13
: obj) =>
({def} Ded ([vinev_14
: that v_13 E prime2 (.thelaw_1, u_8)) =>
({def} Simp1 (vinev_14
Iff1 v_13 Ui Ug ([u_19
: obj) =>
({def} Dediff ([hyp1_20
: that u_19 E prime2
(.thelaw_1, u_8)) =>
({def} Simp1 (hyp1_20
Iff1 u_19 Ui u_8
Compax Usc (.thelaw_1
(u_8))) Conj
Negintro ([hyp3_22
: that u_19 = .thelaw_1
(u_8)) =>
({def} Subs (Eqsymm
(hyp3_22), [(v_24
: obj) =>
({def} v_24
E Usc (.thelaw_1
(u_8)) : prop)], Inusc2
(.thelaw_1 (u_8))) Mp
Simp2 (hyp1_20
Iff1 u_19 Ui u_8
Compax Usc (.thelaw_1
(u_8))) : that
??)]) : that
(u_19 E u_8) & ~ (u_19
= .thelaw_1 (u_8))), [(hyp2_20
: that (u_19 E u_8) & ~ (u_19
= .thelaw_1 (u_8))] =>
({def} Simp1 (hyp2_20) Conj
Negintro ([hyp4_23

```

```

      : that u_19 E Usc
      (.thelaw_1 (u_8))) =>
      ({def} Inusc1
      (hyp4_23) Mp
      Simp2 (hyp2_20) : that
      ??)]) Iff2
    u_19 Ui u_8 Compax
    Usc (.thelaw_1 (u_8)) : that
    u_19 E u_8 Complement
    Usc (.thelaw_1 (u_8)))] : that
    (u_19 E prime2 (.thelaw_1, u_8)) ==
    (u_19 E u_8) & ~ (u_19
    = .thelaw_1 (u_8)))] Mp
    v_13 Ui Simp1 (usinev_9
    Iff1 u_8 Ui Scthm (.M_1)) : that
    v_13 E .M_1)] : that
    (v_13 E prime2 (.thelaw_1, u_8)) ->
    v_13 E .M_1)] Conj (Isset
    (prime2 (.thelaw_1, u_8)) Fixform
    Separation3 (Refleq (prime2
    (.thelaw_1, u_8)))) Conj
    Misset_1) Iff2 prime2 (.thelaw_1, u_8) Ui
    Scthm (.M_1) : that prime2
    (.thelaw_1, u_8) E Sc (.M_1)))] : that
    (u_8 E Sc (.M_1)) -> prime2
    (.thelaw_1, u_8) E Sc (.M_1)))] Conj
    Ug ([u_8 : obj) =>
    ({def} Ug ([v_9 : obj) =>
    ({def} Ded ([hyp_10 : that
    (u_8 <=< Sc (.M_1)) & v_9
    E u_8) =>
    ({def} (((u_8 Intersection
    v_9) <=< .M_1) Fixform Ug
    ([w_14 : obj) =>
    ({def} Ded ([hyp2_15
    : that w_14 E u_8 Intersection
    v_9) =>
    ({def} Simp2 (hyp_10) Mp

```

```

v_9 Ui hyp2_15 Iff1
w_14 Ui Simp2 (hyp_10) Mp
u_8 Intax v_9 Mp w_14
Ui Simp1 (Simp2 (hyp_10) Mp
v_9 Ui Simp1 (Simp1
(hyp_10)) Iff1 v_9
Ui Scthm (.M_1)) : that
w_14 E .M_1))) : that
(w_14 E u_8 Intersection
v_9) -> w_14 E .M_1))) Conj
(Isset (u_8 Intersection
v_9) Fixform Separation3
(Refleq (u_8 Intersection
v_9))) Conj Misset_1) Iff2
(u_8 Intersection v_9) Ui
Scthm (.M_1) : that (u_8
Intersection v_9) E Sc (.M_1)))) : that
((u_8 <= Sc (.M_1)) & v_9
E u_8) -> (u_8 Intersection
v_9) E Sc (.M_1)))) : that
Forall ([ (x'_9 : obj) =>
({def} ((u_8 <= Sc (.M_1)) & x'_9
E u_8) -> (u_8 Intersection
x'_9) E Sc (.M_1) : prop)))])) : that
thetachain1 (.M_1, .thelaw_1, Sc
(.M_1))))]

```

```

thetasc2 : [(M_1 : obj), (Misset_1
: that Isset (.M_1)), (.thelaw_1
: [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
: [(S_2 : obj), (subsestev_2 : that
.S_2 <= .M_1), (inev_2 : that
Exists [(x_4 : obj) =>
({def} x_4 E .S_2 : prop)])) =>
(--- : that .thelaw_1 (.S_2) E .S_2))] =>
(--- : that thetachain1 (.M_1, .thelaw_1, Sc
(.M_1))))]

```

```

{move 0}

>>> open

{move 2}

>>> define thetascm : thetascm2 Misset, thelawchooses

thetascm : [
  ({def} Misset thetascm2 thelawchooses
   : that thetachain1 (M, [(S''_2
    : obj) =>
    ({def} thelaw (S''_2) : obj)], Sc
    (M))))]

thetascm : that thetachain1 (M, [(S''_2
  : obj) =>
  ({def} thelaw (S''_2) : obj)], Sc
  (M))

{move 1}
end Lestrade execution

```

Here we have proved that $\mathcal{P}(M)$ is a Θ -chain.

Notice that we take this theorem down to move 0 then bring it back up and define a new move 1 theorem with the same content in terms of the move 0 concept. This prevents future references to this theorem from expanding to the very large term appearing above.

```

begin Lestrade execution

```

```

>>> clearcurrent

{move 2}

>>> define Thetachain : Set ((Sc \
    (Sc M)), thetachain)

Thetachain : Sc (Sc (M)) Set thetachain

Thetachain : obj

{move 1}

>>> open

    {move 3}

    >>> declare C obj

    C : obj

    {move 3}

    >>> open

        {move 4}

        >>> declare hyp1 that thetachain \
            C

```



```
hyp1 : that thetachain (C)
```

```
{move 4}
```

```
>>> declare hyp2 that C E Thetachain
```

```
hyp2 : that C E Thetachain
```

```
{move 4}
```

```
>>> define line1 hyp1 : Iff2 \
      (Simp1 (Simp2 hyp1), Ui C Scthm \
      Sc M)
```

```
line1 : [(hyp1_1 : that thetachain
      (C)) =>
      ({def} Simp1 (Simp2 (hyp1_1)) Iff2
      C Ui Scthm (Sc (M)) : that
      C E Sc (Sc (M)))]
```

```
line1 : [(hyp1_1 : that thetachain
      (C)) => (--- : that C E Sc
      (Sc (M)))]
```

```
{move 3}
```

```
>>> define line2 hyp1 : Fixform \
      (C E Thetachain, Iff2 (Conj \
      (line1 hyp1, hyp1), Ui (C, Separation \
      ((Sc (Sc M)), thetachain))))
```

```

line2 : [(hyp1_1 : that thetachain
(C)) =>
({def} (C E Thetachain) Fixform
line1 (hyp1_1) Conj hyp1_1
Iff2 C Ui Sc (Sc (M)) Separation
thetachain : that C E Thetachain)]

line2 : [(hyp1_1 : that thetachain
(C)) => (--- : that C E Thetachain)]

```

```

{move 3}

```

```

>>> define line3 hyp2 : Simp2 \
(Iff1 (hyp2, Ui (C, Separation \
((Sc (Sc M)), thetachain))))

```

```

line3 : [(hyp2_1 : that C E Thetachain) =>
({def} Simp2 (hyp2_1 Iff1
C Ui Sc (Sc (M)) Separation
thetachain) : that thetachain
(C)))]

```

```

line3 : [(hyp2_1 : that C E Thetachain) =>
(--- : that thetachain (C)))]

```

```

{move 3}

```

```

>>> close

```

```

{move 3}

```

```

>>> define line4 C : Dediff line3, line2

```

```

line4 : [(C_1 : obj) =>
  ({def} Dediff [(hyp2_11
    : that C_1 E Thetachain) =>
    ({def} Simp2 (hyp2_11 Iff1
      C_1 Ui Sc (Sc (M)) Separation
      thetachain) : that thetachain
      (C_1))], [(hyp1_11
        : that thetachain (C_1)) =>
        ({def} (C_1 E Thetachain) Fixform
          Simp1 (Simp2 (hyp1_11)) Iff2
          C_1 Ui Scthm (Sc (M)) Conj
          hyp1_11 Iff2 C_1 Ui Sc (Sc
            (M)) Separation thetachain
            : that C_1 E Thetachain)]) : that
          (C_1 E Thetachain) == thetachain
          (C_1)))]

```

```

line4 : [(C_1 : obj) => (---
  : that (C_1 E Thetachain) ==
  thetachain (C_1))]

```

```

{move 2}

```

```

>>> close

```

```

{move 2}

```

```

>>> define Thetachainax : Ug line4

```

```

Thetachainax : Ug [(C_2 : obj) =>
  ({def} Dediff [(hyp2_3 : that
    C_2 E Thetachain) =>

```

```

      ({def} Simp2 (hyp2_3 Iff1 C_2
      Ui Sc (Sc (M)) Separation
      thetachain) : that thetachain
      (C_2))] , [(hyp1_3 : that
      thetachain (C_2)) =>
      ({def} (C_2 E Thetachain) Fixform
      Simp1 (Simp2 (hyp1_3)) Iff2
      C_2 Ui Scthm (Sc (M)) Conj
      hyp1_3 Iff2 C_2 Ui Sc (Sc (M)) Separation
      thetachain : that C_2 E Thetachain)]) : that
      (C_2 E Thetachain) == thetachain
      (C_2))])

```

```

Thetachainax : that Forall ([ (x'_15
: obj) =>
({def} (x'_15 E Thetachain) ==
thetachain (x'_15) : prop)])

```

```

      {move 1}
end Lestrade execution

```

We prove that the collection of all Θ -chains is a set, and in particular a subset of $\mathcal{P}^2(M)$.

We now define the Θ -chain which implements the desired well-ordering (though we have to verify subsequently that that is what it is).

```

begin Lestrade execution

```

```

>>> define Mbold1 : Intersection Thetachain \
      Sc M

```

```

Mbold1 : [
      ({def} Thetachain Intersection
      Sc (M) : obj)]

```

```

Mbold1 : obj

{move 1}

>>> close

{move 1}

>>> define Mbold2 Misset thelawchooses \
      : Mbold1

Mbold2 : [(M_1 : obj), (Misset_1
      : that Isset (M_1)), (.thelaw_1
      : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
      : [(S_2 : obj), (subsestev_2 : that
      .S_2 <=<= .M_1), (inev_2 : that
      Exists ([(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])) =>
      (--- : that .thelaw_1 (.S_2) E .S_2))] =>
      ({def} (Sc (Sc (M_1)) Set [(C_2
      : obj) =>
      ({def} thetachain1 (M_1, .thelaw_1, C_2) : prop)])) Intersection
      Sc (M_1) : obj)]

Mbold2 : [(M_1 : obj), (Misset_1
      : that Isset (M_1)), (.thelaw_1
      : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
      : [(S_2 : obj), (subsestev_2 : that
      .S_2 <=<= .M_1), (inev_2 : that
      Exists ([(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])) =>
      (--- : that .thelaw_1 (.S_2) E .S_2))] =>

```

```

      (--- : obj)]

{move 0}

>>> open

      {move 2}

>>> define Mbold : Mbold2 Misset, thelawchooses

Mbold : [
  ({def} Misset Mbold2 thelawchooses
   : obj)]

Mbold : obj

      {move 1}
end Lestrade execution

```

We now have the tedious task of directly verifying that \mathbf{M} is a Θ -chain, which Zermelo dismisses as a side remark!

We note that Zermelo's text suggests that we should prove that any intersection of Θ -chains is a Θ -chain, but it appears that the only case of this we need is that \mathbf{M} itself is a Θ -chain.

```

begin Lestrade execution

      >>> clearcurrent

{move 2}

      >>> declare C obj

```

```
C : obj
```

```
{move 2}
```

```
>>> declare D obj
```

```
D : obj
```

```
{move 2}
```

```
>>> open
```

```
{move 3}
```

```
>>> define Mboldax1 : Intax Thetachain \  
      Sc M
```

```
Mboldax1 : [  
  ({def} Thetachain Intax Sc (M) : that  
  (Sc (M) E Thetachain) ->  
  Forall ([ (x''_3 : obj) =>  
    ({def} (x''_3 E Thetachain  
    Intersection Sc (M)) ==  
    Forall ([ (B1_5 : obj) =>  
      ({def} (B1_5 E Thetachain) ->  
      x''_3 E B1_5 : prop])) : prop)))]
```

```
Mboldax1 : that (Sc (M) E Thetachain) ->  
  Forall ([ (x''_3 : obj) =>  
    ({def} (x''_3 E Thetachain
```

```

Intersection Sc (M)) == Forall
  ([ (B1_5 : obj) =>
    ({def} (B1_5 E Thetachain) ->
      x''_3 E B1_5 : prop)]) : prop)])

{move 2}

>>> define line1 : Ui Sc M Thetachainax

line1 : Sc (M) Ui Thetachainax

line1 : that (Sc (M) E Thetachain) ==
  thetachain (Sc (M))

{move 2}

>>> define line2 : Iff2 thetascm \
  line1

line2 : [
  ({def} thetascm Iff2 line1 : that
    Sc (M) E Thetachain)]

line2 : that Sc (M) E Thetachain

{move 2}

>>> close

{move 2}

```



```

>>> define Mboldax : Fixform (Forall \
  [C => (C E Mbold) == Forall [D => \
    (D E Thetachain) -> C E D]], Mp \
  line2 Mboldax1)

Mboldax : [
  ({def} Forall ([C_9 : obj) =>
    ({def} (C_9 E Mbold) == Forall
      ([D_11 : obj) =>
        ({def} (D_11 E Thetachain) ->
          C_9 E D_11 : prop)])) : prop)]) Fixform
  thetascm Iff2 Sc (M) Ui Thetachainax
  Mp Thetachain Intax Sc (M) : that
  Forall ([C_8 : obj) =>
    ({def} (C_8 E Mbold) == Forall
      ([D_10 : obj) =>
        ({def} (D_10 E Thetachain) ->
          C_8 E D_10 : prop)])) : prop]]))

Mboldax : that Forall ([C_8 : obj) =>
  ({def} (C_8 E Mbold) == Forall
    ([D_10 : obj) =>
      ({def} (D_10 E Thetachain) ->
        C_8 E D_10 : prop)])) : prop]])

{move 1}
end Lestrade execution

```

Above, we develop the most convenient definition of the extension of **M**. I am not sure it is actually used much (though it is used at least once). I believe that development of **Separation4** caused separation axioms for particular constructions to be used much less.

```

begin Lestrade execution

    >>> clearcurrent

{move 2}

    >>> open

    {move 3}

    >>> declare F obj

    F : obj

    {move 3}

    >>> open

    {move 4}

    >>> declare ftheta that F E Thetachain

    ftheta : that F E Thetachain

    {move 4}

    >>> define line1 ftheta : Iff1 \
        (ftheta, Ui F Thetachainax)

    line1 : [(ftheta_1 : that F E Thetachain) =>
        ({def} ftheta_1 Iff1 F Ui

```

```

Thetachainax : that thetachain
(F))]]

line1 : [(ftheta_1 : that F E Thetachain) =>
  (--- : that thetachain (F))]]

{move 3}

>>> define line2 ftheta : Simp1 \
  line1 ftheta

line2 : [(ftheta_1 : that F E Thetachain) =>
  ({def} Simp1 (line1 (ftheta_1)) : that
  M E F)]

line2 : [(ftheta_1 : that F E Thetachain) =>
  (--- : that M E F)]

{move 3}

>>> close

{move 3}

>>> define Linea1 F : Ded line2

Linea1 : [(F_1 : obj) =>
  ({def} Ded ([(ftheta_6 : that
  F_1 E Thetachain) =>
  ({def} Simp1 (ftheta_6 Iff1
  F_1 Ui Thetachainax) : that

```

```

      M E F_1])) : that (F_1
E Thetachain) -> M E F_1)]

Linea1 : [(F_1 : obj) => (---
      : that (F_1 E Thetachain) ->
      M E F_1)]

{move 2}

>>> close

{move 2}

>>> define Lineb1 : Ug Linea1

Lineb1 : Ug ([(F_2 : obj) =>
      ({def} Ded ([(ftheta_3 : that
      F_2 E Thetachain) =>
      ({def} Simp1 (ftheta_3 Iff1
      F_2 Ui Thetachainax) : that
      M E F_2)))] : that (F_2 E Thetachain) ->
      M E F_2)])

Lineb1 : that Forall ([(x'_10 : obj) =>
      ({def} (x'_10 E Thetachain) ->
      M E x'_10 : prop)])

{move 1}

>>> define Line1 : Iff2 (Lineb1, Ui \
      M Mboldax)

```

```

Line1 : [
  ({def} Lineb1 Iff2 M Ui Mboldax
   : that M E Mbold)]

```

```

Line1 : that M E Mbold

```

```

{move 1}

```

```

>>> clearcurrent

```

```

{move 2}
end Lestrade execution

```

Here is the first component of the proof that \mathbf{M} is a Θ -chain.

```

begin Lestrade execution

```

```

>>> open

```

```

{move 3}

```

```

>>> open

```

```

{move 4}

```

```

>>> declare A obj

```

```

A : obj

```

```

{move 4}

```

```

>>> open

{move 5}

>>> declare ainev that A E Mbold

ainev : that A E Mbold

{move 5}

>>> define line1 ainev : Mp \
      (Iff2 (thetascm, Ui Sc \
      M Thetachainax), Ui Sc M, Iff1 \
      (ainev, Ui A Mboldax))

line1 : [(ainev_1 : that
      A E Mbold) =>
      ({def} thetascm Iff2 Sc
      (M) Ui Thetachainax Mp
      Sc (M) Ui ainev_1 Iff1
      A Ui Mboldax : that A E Sc
      (M)))]

line1 : [(ainev_1 : that
      A E Mbold) => (--- : that
      A E Sc (M)))]

{move 4}

>>> close

```

```
{move 4}
```

```
>>> define line2 A : Ded line1
```

```
line2 : [(A_1 : obj) =>
  ({def} Ded ([(ainev_2
    : that A_1 E Mbold) =>
    ({def} thetascm Iff2 Sc
      (M) Ui Thetachainax Mp
      Sc (M) Ui ainev_2 Iff1
      A_1 Ui Mboldax : that A_1
      E Sc (M)))] : that
  (A_1 E Mbold) -> A_1 E Sc
  (M))]
```

```
line2 : [(A_1 : obj) => (---
  : that (A_1 E Mbold) ->
  A_1 E Sc (M))]
```

```
{move 3}
```

```
>>> close
```

```
{move 3}
```

```
>>> define Line3 : Ug line2
```

```
Line3 : Ug ([(A_2 : obj) =>
  ({def} Ded ([(ainev_3 : that
    A_2 E Mbold) =>
    ({def} thetascm Iff2 Sc (M) Ui
      Thetachainax Mp Sc (M) Ui
```

```

      ainev_3 Iff1 A_2 Ui Mboldax
      : that A_2 E Sc (M))]] : that
      (A_2 E Mbold) -> A_2 E Sc (M))]]

```

```

Line3 : that Forall ([ (x'_11
      : obj) =>
      ({def} (x'_11 E Mbold) ->
      x'_11 E Sc (M) : prop)])

```

```

{move 2}

```

```

>>> close

```

```

{move 2}

```

```

>>> define Line4 : Fixform ((Mbold) <=& \
      Sc M, Conj (Line3, Conj (Inhabited \
      Line1, Sc2 M)))

```

```

Line4 : [
      ({def} (Mbold <=& Sc (M)) Fixform
      Ug ([ (A_3 : obj) =>
      ({def} Ded ([ (ainev_4 : that
      A_3 E Mbold) =>
      ({def} thetascm Iff2 Sc (M) Ui
      Thetachainax Mp Sc (M) Ui
      ainev_4 Iff1 A_3 Ui Mboldax
      : that A_3 E Sc (M))]] : that
      (A_3 E Mbold) -> A_3 E Sc (M))]] Conj
      Inhabited (Line1) Conj Sc2 (M) : that
      Mbold <=& Sc (M))]

```

```

Line4 : that Mbold <=& Sc (M)

```



```
{move 1}
```

```
>>> clearcurrent
```

```
{move 2}
```

```
end Lestrade execution
```

Here is the second component of the proof that \mathbf{M} is a Θ -chain.

```
begin Lestrade execution
```

```
>>> open
```

```
{move 3}
```

```
>>> declare F obj
```

```
F : obj
```

```
{move 3}
```

```
>>> open
```

```
{move 4}
```

```
>>> declare finmbold that F E (Mbold)
```

```
finmbold : that F E Mbold
```

```
{move 4}
```

```
>>> open
```

```
{move 5}
```

```
>>> declare G obj
```

```
G : obj
```

```
{move 5}
```

```
>>> open
```

```
{move 6}
```

```
>>> declare gtheta that \  
      G E Thetachain
```

```
gtheta : that G E Thetachain
```

```
{move 6}
```

```
>>> define line1 gtheta \  
      : Ui (F, Simp1 Simp2 \  
      Simp2 Iff1 (gtheta, Ui \  
      G Thetachainax))
```

```
line1 : [(gtheta_1 : that  
      G E Thetachain) =>  
      ({def} F Ui Simp1 (Simp2
```

```

(Simp2 (gtheta_1 Iff1
G Ui Thetachainax))) : that
(F E G) -> prime2
(thelaw, F) E G)]

line1 : [(gtheta_1 : that
G E Thetachain) =>
(--- : that (F E G) ->
prime2 (thelaw, F) E G)]

{move 5}

>>> define line2 gtheta \
: Mp (gtheta, Ui (G, Iff1 \
(finmbold, Ui F Mboldax)))

line2 : [(gtheta_1 : that
G E Thetachain) =>
({def} gtheta_1 Mp
G Ui finmbold Iff1 F Ui
Mboldax : that F E G)]

line2 : [(gtheta_1 : that
G E Thetachain) =>
(--- : that F E G)]

{move 5}

>>> define line3 gtheta \
: Mp line2 gtheta line1 \
gtheta

```

```

line3 : [(gtheta_1 : that
  G E Thetachain) =>
  ({def} line2 (gtheta_1) Mp
  line1 (gtheta_1) : that
  prime2 (thelaw, F) E G)]

line3 : [(gtheta_1 : that
  G E Thetachain) =>
  (--- : that prime2
  (thelaw, F) E G)]

{move 5}

>>> close

{move 5}

>>> define line4 G : Ded line3

line4 : [(G_1 : obj) =>
  ({def} Ded ([gtheta_12
    : that G_1 E Thetachain) =>
    ({def} gtheta_12 Mp
    G_1 Ui finmbold Iff1
    F Ui Mboldax Mp F Ui
    Simp1 (Simp2 (Simp2
    (gtheta_12 Iff1 G_1
    Ui Thetachainax))) : that
    prime2 (thelaw, F) E G_1)]) : that
  (G_1 E Thetachain) ->
  prime2 (thelaw, F) E G_1)]

line4 : [(G_1 : obj) =>

```

```

      (--- : that (G_1 E Thetachain) ->
prime2 (thelaw, F) E G_1)]

{move 4}

>>> close

{move 4}

>>> define line5 finmbold : Ug \
line4

line5 : [(finmbold_1 : that
F E Mbold) =>
({def} Ug ([G_2 : obj) =>
({def} Ded ([gtheta_3
: that G_2 E Thetachain) =>
({def} gtheta_3 Mp
G_2 Ui finmbold_1 Iff1
F Ui Mboldax Mp F Ui
Simp1 (Simp2 (Simp2
(gtheta_3 Iff1 G_2
Ui Thetachainax))) : that
prime2 (thelaw, F) E G_2)]) : that
(G_2 E Thetachain) ->
prime2 (thelaw, F) E G_2)]) : that
Forall ([x'_14 : obj) =>
({def} (x'_14 E Thetachain) ->
prime2 (thelaw, F) E x'_14
: prop)]))]

line5 : [(finmbold_1 : that
F E Mbold) => (--- : that
Forall ([x'_14 : obj) =>

```

```

({def} (x'_14 E Thetachain) ->
prime2 (thelaw, F) E x'_14
: prop)))]

```

```

{move 3}

```

```

>>> define line6 finmbold : Iff2 \
      (line5 finmbold, Ui (prime \
      F, Mboldax))

```

```

line6 : [(finmbold_1 : that
      F E Mbold) =>
      ({def} line5 (finmbold_1) Iff2
      prime (F) Ui Mboldax : that
      prime (F) E Mbold)]

```

```

line6 : [(finmbold_1 : that
      F E Mbold) => (--- : that
      prime (F) E Mbold)]

```

```

{move 3}

```

```

>>> close

```

```

{move 3}

```

```

>>> define line7 F : Ded line6

```

```

line7 : [(F_1 : obj) =>
      ({def} Ded [(finmbold_15
      : that F_1 E Mbold) =>
      ({def} Ug [(G_18 : obj) =>

```

```

      ({def} Ded ([gtheta_19
        : that G_18 E Thetachain) =>
      ({def} gtheta_19 Mp
        G_18 Ui finmbold_15
        Iff1 F_1 Ui Mboldax
        Mp F_1 Ui Simp1 (Simp2
          (Simp2 (gtheta_19
            Iff1 G_18 Ui Thetachainax))) : that
        prime2 (thelaw, F_1) E G_18)]) : that
      (G_18 E Thetachain) ->
      prime2 (thelaw, F_1) E G_18)]) Iff2
    prime (F_1) Ui Mboldax : that
    prime (F_1) E Mbold)]) : that
  (F_1 E Mbold) -> prime (F_1) E Mbold)]

```

```

line7 : [(F_1 : obj) => (---
  : that (F_1 E Mbold) -> prime
  (F_1) E Mbold)]

```

```

{move 2}

```

```

>>> close

```

```

{move 2}

```

```

>>> define Linea8 : Ug line7

```

```

Linea8 : Ug ([F_2 : obj) =>
  ({def} Ded ([finmbold_3 : that
    F_2 E Mbold) =>
  ({def} Ug ([G_5 : obj) =>
    ({def} Ded ([gtheta_6
      : that G_5 E Thetachain) =>
    ({def} gtheta_6 Mp G_5

```

```

      Ui finmbold_3 Iff1 F_2
      Ui Mboldax Mp F_2 Ui Simp1
      (Simp2 (Simp2 (gtheta_6
      Iff1 G_5 Ui Thetachainax))) : that
      prime2 (thelaw, F_2) E G_5]] : that
      (G_5 E Thetachain) -> prime2
      (thelaw, F_2) E G_5]] Iff2
      prime (F_2) Ui Mboldax : that
      prime (F_2) E Mbold))] : that
      (F_2 E Mbold) -> prime (F_2) E Mbold))]

```

```

Linea8 : that Forall ([ (x'_17 : obj) =>
      ({def} (x'_17 E Mbold) -> prime
      (x'_17) E Mbold : prop)])

```

```

{move 1}

```

```

>>> save

```

```

{move 2}

```

```

>>> close

```

```

{move 1}

```

```

>>> define Lineb8 Misset thelawchooses \
      : Linea8

```

```

Lineb8 : [(M_1 : obj), (Misset_1
      : that Isset (M_1)), (thelaw_1
      : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
      : [(S_2 : obj), (subsestev_2 : that
      .S_2 <= .M_1), (inev_2 : that

```



```

Exists ([x_4 : obj) =>
  ({def} x_4 E .S_2 : prop]])) =>
  (--- : that .thelaw_1 (.S_2) E .S_2)]) =>
({def} Ug ([F_4 : obj) =>
  ({def} Ded ([finmbold_5 : that
    F_4 E Misset_1 Mbold2 thelawchooses_1) =>
    ({def} Ug ([G_7 : obj) =>
      ({def} Ded ([gtheta_8
        : that G_7 E Sc (Sc (.M_1)) Set
        [(C_11 : obj) =>
          ({def} thetachain1
            (.M_1, .thelaw_1, C_11) : prop]])) =>
        ({def} gtheta_8 Mp G_7
          Ui finmbold_5 Iff1 F_4
          Ui Forall ([C_15 : obj) =>
            ({def} (C_15 E Misset_1
              Mbold2 thelawchooses_1) ==
              Forall ([D_17 : obj) =>
                ({def} (D_17 E Sc
                  (Sc (.M_1)) Set
                  [(C_20 : obj) =>
                    ({def} thetachain1
                      (.M_1, .thelaw_1, C_20) : prop]])) ->
                    C_15 E D_17 : prop]])) : prop]])) Fixform
          Misset_1 thetascm2 thelawchooses_1
          Iff2 Sc (.M_1) Ui Ug
          [(C_18 : obj) =>
            ({def} Dediff ([hyp2_19
              : that C_18 E Sc
              (Sc (.M_1)) Set
              [(C_22 : obj) =>
                ({def} thetachain1
                  (.M_1, .thelaw_1, C_22) : prop]])) =>
                ({def} Simp2 (hyp2_19
                  Iff1 C_18 Ui Sc (Sc
                    (.M_1)) Separation
                    [(C_23 : obj) =>
                      ({def} thetachain1

```

```

      (.M_1, .thelaw_1, C_23) : prop]]) : that
thetachain1 (.M_1, .thelaw_1, C_18))], [(hyp1_19
: that thetachain1
(.M_1, .thelaw_1, C_18)) =>
({def} (C_18 E Sc
(Sc (.M_1)) Set
[(C_22 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_22) : prop)]) Fixform
Simp1 (Simp2 (hyp1_19)) Iff2
C_18 Ui Scthm (Sc
(.M_1)) Conj hyp1_19
Iff2 C_18 Ui Sc (Sc
(.M_1)) Separation
[(C_23 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_23) : prop))] : that
C_18 E Sc (Sc (.M_1)) Set
[(C_21 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_21) : prop))]]]) : that
(C_18 E Sc (Sc (.M_1)) Set
[(C_21 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_21) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_18))]) Mp
(Sc (Sc (.M_1)) Set
[(C_17 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_17) : prop)]) Intax
Sc (.M_1) Mp F_4 Ui Simp1
(Simp2 (Simp2 (gtheta_8
Iff1 G_7 Ui Ug [(C_16
: obj) =>
({def} Dediff [(hyp2_17
: that C_16 E Sc
(Sc (.M_1)) Set
[(C_20 : obj) =>

```

```

      ({def} thetachain1
      (.M_1, .thelaw_1, C_20) : prop)]) =>
({def} Simp2 (hyp2_17
Iff1 C_16 Ui Sc (Sc
(.M_1)) Separation
[(C_21 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_21) : prop)]) : that
thetachain1 (.M_1, .thelaw_1, C_16))], [(hyp1_17
: that thetachain1
(.M_1, .thelaw_1, C_16)) =>
({def} (C_16 E Sc
(Sc (.M_1)) Set
[(C_20 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_20) : prop)]) Fixform
Simp1 (Simp2 (hyp1_17)) Iff2
C_16 Ui Scthm (Sc
(.M_1)) Conj hyp1_17
Iff2 C_16 Ui Sc (Sc
(.M_1)) Separation
[(C_21 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_21) : prop)] : that
C_16 E Sc (Sc (.M_1)) Set
[(C_19 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_19) : prop)]]) : that
(C_16 E Sc (Sc (.M_1)) Set
[(C_19 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_19) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_16))])]) : that
prime2 (.thelaw_1, F_4) E G_7)]) : that
(G_7 E Sc (Sc (.M_1)) Set
[(C_10 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_10) : prop)]) ->
prime2 (.thelaw_1, F_4) E G_7)]) Iff2

```

```

prime2 (.thelaw_1, F_4) Ui
Forall ([C_9 : obj) =>
  ({def} (C_9 E Misset_1 Mbold2
thelawchooses_1) == Forall
  ([D_11 : obj) =>
    ({def} (D_11 E Sc (Sc
      (.M_1)) Set [(C_14
        : obj) =>
          ({def} thetachain1
            (.M_1, .thelaw_1, C_14) : prop)]) ->
      C_9 E D_11 : prop)]) : prop)]) Fixform
Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug ([C_12
  : obj) =>
  ({def} Dediff ([hyp2_13
    : that C_12 E Sc (Sc (.M_1)) Set
    [(C_16 : obj) =>
      ({def} thetachain1
        (.M_1, .thelaw_1, C_16) : prop)]) =>
    ({def} Simp2 (hyp2_13
      Iff1 C_12 Ui Sc (Sc (.M_1)) Separation
      [(C_17 : obj) =>
        ({def} thetachain1
          (.M_1, .thelaw_1, C_17) : prop)]) : that
        thetachain1 (.M_1, .thelaw_1, C_12)]), [(hyp1_13
          : that thetachain1 (.M_1, .thelaw_1, C_12)) =>
          ({def} (C_12 E Sc (Sc
            (.M_1)) Set [(C_16
              : obj) =>
                ({def} thetachain1
                  (.M_1, .thelaw_1, C_16) : prop)]) Fixform
          Simp1 (Simp2 (hyp1_13)) Iff2
          C_12 Ui Scthm (Sc (.M_1)) Conj
          hyp1_13 Iff2 C_12 Ui Sc
          (Sc (.M_1)) Separation
          [(C_17 : obj) =>
            ({def} thetachain1
              (.M_1, .thelaw_1, C_17) : prop)] : that

```

```

C_12 E Sc (Sc (.M_1)) Set
[(C_15 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_15) : prop))]] : that
(C_12 E Sc (Sc (.M_1)) Set
[(C_15 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_15) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_12))]] Mp
(Sc (Sc (.M_1)) Set [(C_11
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_11) : prop)]) Intax
Sc (.M_1) : that prime2 (.thelaw_1, F_4) E Misset_1
Mbold2 thelawchooses_1))] : that
(F_4 E Misset_1 Mbold2 thelawchooses_1) ->
prime2 (.thelaw_1, F_4) E Misset_1
Mbold2 thelawchooses_1))] : that
Forall ([ (x'_4 : obj) =>
  ({def} (x'_4 E Misset_1 Mbold2
thelawchooses_1) -> prime2 (.thelaw_1, x'_4) E Misset_1
Mbold2 thelawchooses_1 : prop))]))]

```

```

Lineb8 : [(M_1 : obj), (Misset_1
: that Isset (M_1)), (.thelaw_1
: [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
: [(S_2 : obj), (subsevev_2 : that
.S_2 <= M_1), (inev_2 : that
Exists ([ (x_4 : obj) =>
  ({def} x_4 E .S_2 : prop))])]) =>
(--- : that .thelaw_1 (.S_2) E .S_2))] =>
(--- : that Forall ([ (x'_4 : obj) =>
  ({def} (x'_4 E Misset_1 Mbold2
thelawchooses_1) -> prime2 (.thelaw_1, x'_4) E Misset_1
Mbold2 thelawchooses_1 : prop))]))]

```

{move 0}

```

>>> open

{move 2}

>>> define Line8 : Lineb8 Misset, thelawchooses

Line8 : [
  ({def} Misset Lineb8 thelawchooses
   : that Forall ([x'_2 : obj) =>
    ({def} (x'_2 E Misset Mbold2
    thelawchooses) -> prime2 ([S'_5
      : obj) =>
      ({def} thelaw (S'_5) : obj)], x'_2) E Misset
      Mbold2 thelawchooses : prop)]))]

Line8 : that Forall ([x'_2 : obj) =>
  ({def} (x'_2 E Misset Mbold2 thelawchooses) ->
  prime2 ([S'_5 : obj) =>
    ({def} thelaw (S'_5) : obj)], x'_2) E Misset
    Mbold2 thelawchooses : prop)])

{move 1}
end Lestrade execution

```

Here is the third component of the proof that \mathbf{M} is a Θ -chain. Note the importance of preventing definitional expansion here!

```

begin Lestrade execution

```

```

>>> open

```

```

{move 3}

```

```
>>> declare H obj
```

```
H : obj
```

```
{move 3}
```

```
>>> open
```

```
{move 4}
```

```
>>> declare J obj
```

```
J : obj
```

```
{move 4}
```

```
>>> open
```

```
{move 5}
```

```
>>> declare thehyp that (H <= \
      Mbold) & J E H
```

```
thehyp : that (H <= Mbold) & J E H
```

```
{move 5}
```

```
>>> open
```

```
{move 6}
```

```
>>> declare K obj
```

```
K : obj
```

```
{move 6}
```

```
>>> open
```

```
{move 7}
```

```
>>> declare ktheta that \  
      K E Thetachain
```

```
ktheta : that K E Thetachain
```

```
{move 7}
```

```
>>> define line1 ktheta \  
      : Iff1 (ktheta, Ui \  
      K Thetachainax)
```

```
line1 : [(ktheta_1  
      : that K E Thetachain) =>  
      ({def} ktheta_1  
      Iff1 K Ui Thetachainax  
      : that thetachain  
      (K))]
```



```

line1 : [(ktheta_1
          : that K E Thetachain) =>
          (--- : that thetchain
              (K)))]

```

```

{move 6}

```

```

>>> define line2 ktheta \
      : Ui J, Ui H, Simp2 \
      Simp2 Simp2 line1 ktheta

```

```

line2 : [(ktheta_1
          : that K E Thetachain) =>
          ({def} J Ui H Ui
           Simp2 (Simp2 (Simp2
                        (line1 (ktheta_1)))) : that
           ((H <= K) & J E H) ->
           (H Intersection
            J) E K)]

```

```

line2 : [(ktheta_1
          : that K E Thetachain) =>
          (--- : that ((H <=
                        K) & J E H) ->
           (H Intersection
            J) E K)]

```

```

{move 6}

```

```

>>> open

```

```

{move 8}

```

```
>>> declare P obj
```

```
P : obj
```

```
{move 8}
```

```
>>> open
```

```
{move 9}
```

```
>>> declare phyp \  
      that P E H
```

```
phyp : that P E H
```

```
{move 9}
```

```
>>> define line3 \  
      phyp : Mp (phyp, Ui \  
      P Simp1 Simp1 \  
      thehyp)
```

```
line3 : [(phyp_1  
      : that P E H) =>  
      ({def} phyp_1  
      Mp P Ui Simp1  
      (Simp1 (thepyp)) : that  
      P E Mbold)]
```

```
line3 : [(phyp_1  
      : that P E H) =>
```

```
(--- : that
P E Mbold)]
```

```
{move 8}
```

```
>>> define line4 \
  phyp : Mp (ktheta, K Ui \
  Iff1 line3 phyp, Ui \
  P Mboldax)
```

```
line4 : [(phyp_1
: that P E H) =>
({def} ktheta
Mp K Ui line3
(phyp_1) Iff1
P Ui Mboldax
: that P E K)]
```

```
line4 : [(phyp_1
: that P E H) =>
(--- : that
P E K)]
```

```
{move 8}
```

```
>>> close
```

```
{move 8}
```

```
>>> define line5 \
  P : Ded line4
```

```

line5 : [(P_1 : obj) =>
  ({def} Ded ([(phyp_8
    : that P_1
    E H) =>
    ({def} ktheta
    Mp K Ui phyp_8
    Mp P_1 Ui Simp1
    (Simp1 (thehyp)) Iff1
    P_1 Ui Mboldax
    : that P_1
    E K)]) : that
  (P_1 E H) ->
  P_1 E K)]

```

```

line5 : [(P_1 : obj) =>
  (--- : that (P_1
  E H) -> P_1 E K)]

```

```
{move 7}
```

```
>>> close
```

```
{move 7}
```

```
>>> define test1 ktheta \
      : Ug line5
```

```

test1 : [(ktheta_1
  : that K E Thetachain) =>
  ({def} Ug ([(P_2
    : obj) =>
    ({def} Ded ([(phyp_3
      : that P_2
      E H) =>

```

```

      ({def} ktheta_1
      Mp K Ui phyp_3
      Mp P_2 Ui Simp1
      (Simp1 (thehyp)) Iff1
      P_2 Ui Mboldax
      : that P_2
      E K)]) : that
      (P_2 E H) ->
      P_2 E K)]) : that
Forall ([ (x'_12
: obj) =>
({def} (x'_12
E H) -> x'_12
E K : prop)))]))

```

```

test1 : [(ktheta_1
: that K E Thetachain) =>
(--- : that Forall
([ (x'_12 : obj) =>
({def} (x'_12
E H) -> x'_12
E K : prop)))]))

```

```

{move 6}

```

```

>>> define test2 ktheta \
      : Inhabited Simp2 thehyp

```

```

test2 : [(ktheta_1
: that K E Thetachain) =>
({def} Inhabited
(Simp2 (thehyp)) : that
Isset (H))]

```

```
test2 : [(ktheta_1
: that K E Thetachain) =>
(--- : that Isset
(H))]
```

```
{move 6}
```

```
>>> define test3 ktheta \
: Inhabited (Mp (Simp2 \
thehyp, line5 J))
```

```
test3 : [(ktheta_1
: that K E Thetachain) =>
({def} Inhabited
(Simp2 (thehyp) Mp
Ded ([ (phyp_15
: that J E H) =>
({def} ktheta_1
Mp K Ui phyp_15
Mp J Ui Simp1
(Simp1 (thehyp)) Iff1
J Ui Mboldax : that
J E K)])) : that
Isset (K))]
```

```
test3 : [(ktheta_1
: that K E Thetachain) =>
(--- : that Isset
(K))]
```

```
{move 6}
```

```
>>> define line6 ktheta \
: Fixform (H <= K, Conj \
```

```

(test1 ktheta, Conj \
(test2 ktheta, test3 \
ktheta)))

```

```

line6 : [(ktheta_1
: that K E Thetachain) =>
({def} (H <= K) Fixform
test1 (ktheta_1) Conj
test2 (ktheta_1) Conj
test3 (ktheta_1) : that
H <= K)]

```

```

line6 : [(ktheta_1
: that K E Thetachain) =>
(--- : that H <=
K)]

```

```

{move 6}

```

```

>>> define line7 ktheta \
: Mp (Conj (line6 \
ktheta, Simp2 thehyp), line2 \
ktheta)

```

```

line7 : [(ktheta_1
: that K E Thetachain) =>
({def} line6 (ktheta_1) Conj
Simp2 (thhyp) Mp
line2 (ktheta_1) : that
(H Intersection
J) E K)]

```

```

line7 : [(ktheta_1

```

```

: that K E Thetachain) =>
(--- : that (H Intersection
J) E K)]

{move 6}

>>> close

{move 6}

>>> define line8 K : Ded \
linea7

line8 : [(K_1 : obj) =>
({def} Ded ([ktheta_6
: that K_1 E Thetachain) =>
({def} ((H <=<=
K_1) Fixform Ug
([P_11 : obj) =>
({def} Ded ([phyp_12
: that P_11
E H) =>
({def} ktheta_6
Mp K_1 Ui phyp_12
Mp P_11 Ui
Simp1 (Simp1
(thehyp)) Iff1
P_11 Ui Mboldax
: that P_11
E K_1])) : that
(P_11 E H) ->
P_11 E K_1))] Conj
Inhabited (Simp2
(thehyp)) Conj
Inhabited (Simp2

```



```

(thehyp) Mp Ded
([ (phyp_14 : that
  J E H) =>
  ({def} ktheta_6
  Mp K_1 Ui phyp_14
  Mp J Ui Simp1
  (Simp1 (thepyp)) Iff1
  J Ui Mboldax : that
  J E K_1))))) Conj
Simp2 (thepyp) Mp
J Ui H Ui Simp2 (Simp2
(Simp2 (ktheta_6
Iff1 K_1 Ui Thetachainax))) : that
(H Intersection
J) E K_1)]) : that
(K_1 E Thetachain) ->
(H Intersection J) E K_1]]

```

```

line8 : [(K_1 : obj) =>
  (--- : that (K_1 E Thetachain) ->
  (H Intersection J) E K_1)]

```

```

{move 5}

```

```

>>> close

```

```

{move 5}

```

```

>>> define line9 thehyp : Ug \
line8

```

```

line9 : [(thepyp_1 : that
  (H <= Mbold) & J E H) =>
  ({def} Ug ([ (K_2 : obj) =>

```

```

({def} Ded ([ktheta_3
: that K_2 E Thetachain) =>
({def} ((H <=<=
K_2) Fixform Ug
([P_8 : obj) =>
  ({def} Ded ([phyp_9
    : that P_8
    E H) =>
    ({def} ktheta_3
    Mp K_2 Ui phyp_9
    Mp P_8 Ui Simp1
    (Simp1 (thehyp_1)) Iff1
    P_8 Ui Mboldax
    : that P_8
    E K_2])) : that
  (P_8 E H) ->
  P_8 E K_2])) Conj
Inhabited (Simp2
(thehyp_1)) Conj
Inhabited (Simp2
(thehyp_1) Mp Ded
([phyp_11 : that
  J E H) =>
  ({def} ktheta_3
  Mp K_2 Ui phyp_11
  Mp J Ui Simp1
  (Simp1 (thehyp_1)) Iff1
  J Ui Mboldax : that
  J E K_2]]))) Conj
Simp2 (thehyp_1) Mp
J Ui H Ui Simp2 (Simp2
(Simp2 (ktheta_3
Iff1 K_2 Ui Thetachainax))) : that
(H Intersection
J) E K_2])) : that
(K_2 E Thetachain) ->
(H Intersection J) E K_2])) : that
Forall ([x'_20 : obj) =>

```

```

      ({def} (x'_20 E Thetachain) ->
      (H Intersection J) E x'_20
      : prop)))]

line9 : [(thehyp_1 : that
      (H <= Mbold) & J E H) =>
      (--- : that Forall ([ (x'_20
      : obj) =>
      ({def} (x'_20 E Thetachain) ->
      (H Intersection J) E x'_20
      : prop)))]

{move 4}

>>> define line10 : Ui (H Intersection \
      J, Mboldax)

line10 : (H Intersection
      J) Ui Mboldax

line10 : that ((H Intersection
      J) E Mbold) == Forall ([ (D_3
      : obj) =>
      ({def} (D_3 E Thetachain) ->
      (H Intersection J) E D_3
      : prop)])

{move 4}

>>> define line11 thehyp : Iff2 \
      (line9 thehyp, line10)

```

```

line11 : [(thehyp_1 : that
  (H <=<= Mbold) & J E H) =>
  ({def} line9 (thehyp_1) Iff2
  line10 : that (H Intersection
  J) E Mbold)]

```

```

line11 : [(thehyp_1 : that
  (H <=<= Mbold) & J E H) =>
  (--- : that (H Intersection
  J) E Mbold)]

```

```

{move 4}

```

```

>>> close

```

```

{move 4}

```

```

>>> define line12 J : Ded line11

```

```

line12 : [(J_1 : obj) =>
  ({def} Ded ([thehyp_21
  : that (H <=<= Mbold) & J_1
  E H) =>
  ({def} Ug ([K_24 : obj) =>
  ({def} Ded ([ktheta_25
  : that K_24 E Thetachain) =>
  ({def} ((H <=<=
  K_24) Fixform Ug
  ([P_30 : obj) =>
  ({def} Ded ([phyp_31
  : that P_30
  E H) =>
  ({def} ktheta_25
  Mp K_24 Ui

```

```

      phyp_31 Mp
      P_30 Ui Simp1
      (Simp1 (thehyp_21)) Iff1
      P_30 Ui Mboldax
      : that P_30
      E K_24)]) : that
      (P_30 E H) ->
      P_30 E K_24)]) Conj
Inhabited (Simp2
(thehyp_21)) Conj
Inhabited (Simp2
(thehyp_21) Mp
Ded ([ (phyp_33
      : that J_1 E H) =>
      ({def} ktheta_25
      Mp K_24 Ui phyp_33
      Mp J_1 Ui Simp1
      (Simp1 (thehyp_21)) Iff1
      J_1 Ui Mboldax
      : that J_1 E K_24)])) Conj
Simp2 (thehyp_21) Mp
J_1 Ui H Ui Simp2
(Simp2 (Simp2 (ktheta_25
Iff1 K_24 Ui Thetachainax))) : that
(H Intersection
J_1) E K_24)]) : that
(K_24 E Thetachain) ->
(H Intersection J_1) E K_24)]) Iff2
(H Intersection J_1) Ui
Mboldax : that (H Intersection
J_1) E Mbold)]) : that
((H <= Mbold) & J_1 E H) ->
(H Intersection J_1) E Mbold)]

```

```

line12 : [(J_1 : obj) => (---
      : that ((H <= Mbold) & J_1
      E H) -> (H Intersection

```

```

J_1) E Mbold)]

{move 3}

>>> close

{move 3}

>>> define line13 H : Ug line12

line13 : [(H_1 : obj) =>
  ({def} Ug ([(J_2 : obj) =>
    ({def} Ded ([(thhyp_3
      : that (H_1 <=< Mbold) & J_2
      E H_1) =>
      ({def} Ug ([(K_5 : obj) =>
        ({def} Ded ([(ktheta_6
          : that K_5 E Thetachain) =>
          ({def} ((H_1 <=<
            K_5) Fixform Ug
            [(P_11 : obj) =>
              ({def} Ded ([(phyp_12
                : that P_11
                E H_1) =>
                ({def} ktheta_6
                Mp K_5 Ui phyp_12
                Mp P_11 Ui
                Simp1 (Simp1
                  (thhyp_3)) Iff1
                  P_11 Ui Mboldax
                  : that P_11
                  E K_5)]) : that
                (P_11 E H_1) ->
                P_11 E K_5)]) Conj
                Inhabited (Simp2

```

```

(thehyp_3)) Conj
Inhabited (Simp2
(thehyp_3) Mp Ded
([(phyp_14 : that
  J_2 E H_1) =>
  ({def} ktheta_6
  Mp K_5 Ui phyp_14
  Mp J_2 Ui Simp1
  (Simp1 (thetahyp_3)) Iff1
  J_2 Ui Mboldax
  : that J_2 E K_5)]))) Conj
Simp2 (thetahyp_3) Mp
J_2 Ui H_1 Ui Simp2
(Simp2 (Simp2 (ktheta_6
Iff1 K_5 Ui Thetachainax))) : that
(H_1 Intersection
J_2) E K_5]] : that
(K_5 E Thetachain) ->
(H_1 Intersection J_2) E K_5]] Iff2
(H_1 Intersection J_2) Ui
Mboldax : that (H_1 Intersection
J_2) E Mbold))] : that
((H_1 <= Mbold) & J_2
E H_1) -> (H_1 Intersection
J_2) E Mbold))] : that
Forall ([(x'_23 : obj) =>
  ({def} ((H_1 <= Mbold) & x'_23
  E H_1) -> (H_1 Intersection
  x'_23) E Mbold : prop)])))]

```

```

line13 : [(H_1 : obj) => (---
: that Forall ([(x'_23 : obj) =>
  ({def} ((H_1 <= Mbold) & x'_23
  E H_1) -> (H_1 Intersection
  x'_23) E Mbold : prop)]))]

```

```
{move 2}
```

```
>>> close
```

```
{move 2}
```

```
>>> define Linea14 : Ug line13
```

```
Linea14 : Ug ([H_2 : obj) =>
  ({def} Ug ([J_3 : obj) =>
    ({def} Ded ([thehyp_4 : that
      (H_2 <=<= Mbold) & J_3 E H_2) =>
      ({def} Ug ([K_6 : obj) =>
        ({def} Ded ([ktheta_7
          : that K_6 E Thetachain) =>
          ({def} ((H_2 <=<=
            K_6) Fixform Ug ([P_12
              : obj) =>
              ({def} Ded ([phyp_13
                : that P_12 E H_2) =>
                ({def} ktheta_7
                  Mp K_6 Ui phyp_13
                  Mp P_12 Ui Simp1
                  (Simp1 (thehyp_4)) Iff1
                  P_12 Ui Mboldax
                  : that P_12 E K_6])) : that
                (P_12 E H_2) ->
                P_12 E K_6])) Conj
                Inhabited (Simp2 (thehyp_4)) Conj
                Inhabited (Simp2 (thehyp_4) Mp
                Ded ([phyp_15 : that
                  J_3 E H_2) =>
                  ({def} ktheta_7
                    Mp K_6 Ui phyp_15
                    Mp J_3 Ui Simp1 (Simp1
                    (thehyp_4)) Iff1
```



```

J_3 Ui Mboldax : that
J_3 E K_6]]))) Conj
Simp2 (thhyp_4) Mp
J_3 Ui H_2 Ui Simp2
(Simp2 (Simp2 (ktheta_7
Iff1 K_6 Ui Thetachainax))) : that
(H_2 Intersection J_3) E K_6]] : that
(K_6 E Thetachain) ->
(H_2 Intersection J_3) E K_6]] Iff2
(H_2 Intersection J_3) Ui
Mboldax : that (H_2 Intersection
J_3) E Mbold))] : that
((H_2 <= Mbold) & J_3 E H_2) ->
(H_2 Intersection J_3) E Mbold))] : that
Forall ([x'_3 : obj) =>
({def} ((H_2 <= Mbold) & x'_3
E H_2) -> (H_2 Intersection
x'_3) E Mbold : prop)))]])

```

```

Linea14 : that Forall ([x'_24 : obj) =>
({def} Forall ([x'_25 : obj) =>
({def} ((x'_24 <= Mbold) & x'_25
E x'_24) -> (x'_24 Intersection
x'_25) E Mbold : prop))] : prop))]

```

```
{move 1}
```

```
>>> save
```

```
{move 2}
```

```
>>> close
```

```
{move 1}
```

```

>>> define Lineb14 Misset thelawchooses \
      : Linea14

Lineb14 : [(M_1 : obj), (Misset_1
  : that Isset (M_1)), (.thelaw_1
  : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
  : [(S_2 : obj), (subsetev_2 : that
    .S_2 <= M_1), (inev_2 : that
    Exists ([(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])) =>
    (--- : that .thelaw_1 (.S_2) E .S_2))] =>
  ({def} Ug ([(H_4 : obj) =>
    ({def} Ug ([(J_5 : obj) =>
      ({def} Ded ([(thehyp_6 : that
        (H_4 <= Misset_1 Mbold2
        thelawchooses_1) & J_5 E H_4) =>
        ({def} Ug ([(K_8 : obj) =>
          ({def} Ded ([(ktheta_9
            : that K_8 E Sc (Sc
            (M_1)) Set [(C_2
              : obj) =>
              ({def} thetachain1
              (M_1, .thelaw_1, C_2) : prop)])) =>
              ({def} ((H_4 <=
                K_8) Fixform Ug ([(P_14
                  : obj) =>
                  ({def} Ded ([(phyp_15
                    : that P_14 E H_4) =>
                    ({def} ktheta_9
                    Mp K_8 Ui phyp_15
                    Mp P_14 Ui Simp1
                    (Simp1 (thehyp_6)) Iff1
                    P_14 Ui Forall
                    ([(C_21 : obj) =>
                      ({def} (C_21
                        E Misset_1

```

```

Mbold2 thelawchooses_1) ==
Forall ([ (D_23
: obj) =>
({def} (D_23
E Sc (Sc
(.M_1)) Set
[(C_2
: obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) ->
C_21 E D_23
: prop)]) : prop)]) Fixform
Misset_1 thetascm2
thelawchooses_1
Iff2 Sc (.M_1) Ui
Ug ([ (C_24
: obj) =>
({def} Dediff
([ (hyp2_25
: that C_24
E Sc (Sc
(.M_1)) Set
[(C_2
: obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) =>
({def} Simp2
(hyp2_25
Iff1 C_24
Ui Sc (Sc
(.M_1)) Separation
[(C_2
: obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1
(.M_1, .thelaw_1, C_24)]), [(hyp1_25
: that thetachain1

```

```

(M_1, .thelaw_1, C_24)) =>
({def} (C_24
E Sc (Sc
(M_1)) Set
[(C_2
: obj) =>
({def} thetachain1
(M_1, .thelaw_1, C_2) : prop)]) Fixform
Simp1 (Simp2
(hyp1_25)) Iff2
C_24 Ui
Scthm (Sc
(M_1)) Conj
hyp1_25
Iff2 C_24
Ui Sc (Sc
(M_1)) Separation
[(C_2
: obj) =>
({def} thetachain1
(M_1, .thelaw_1, C_2) : prop)] : that
C_24 E Sc
(Sc (M_1)) Set
[(C_2
: obj) =>
({def} thetachain1
(M_1, .thelaw_1, C_2) : prop)))] : that
(C_24 E Sc
(Sc (M_1)) Set
[(C_2 : obj) =>
({def} thetachain1
(M_1, .thelaw_1, C_2) : prop)]) ==
thetachain1
(M_1, .thelaw_1, C_24)))] Mp
(Sc (Sc (M_1)) Set
[(C_2 : obj) =>
({def} thetachain1
(M_1, .thelaw_1, C_2) : prop)]) Intax

```

```

      Sc (.M_1) : that
      P_14 E K_8)]) : that
    (P_14 E H_4) ->
      P_14 E K_8)]) Conj
  Inhabited (Simp2 (thehyp_6)) Conj
  Inhabited (Simp2 (thehyp_6) Mp
Ded ([ (phyp_17 : that
      J_5 E H_4) =>
      ({def} ktheta_9
      Mp K_8 Ui phyp_17
      Mp J_5 Ui Simp1 (Simp1
      (thehyp_6)) Iff1
      J_5 Ui Forall ([ (C_23
        : obj) =>
        ({def} (C_23
        E Misset_1 Mbold2
        thelawchooses_1) ==
        Forall ([ (D_25
          : obj) =>
          ({def} (D_25
          E Sc (Sc (.M_1)) Set
          [(C_2 : obj) =>
            ({def} thetachain1
              (.M_1, .thelaw_1, C_2) : prop)]) ->
          C_23 E D_25
          : prop)]) : prop)]) Fixform
  Misset_1 thetascm2
  thelawchooses_1 Iff2
  Sc (.M_1) Ui Ug
  [(C_26 : obj) =>
    ({def} Dediff
    [(hyp2_27
      : that C_26
      E Sc (Sc (.M_1)) Set
      [(C_2 : obj) =>
        ({def} thetachain1
          (.M_1, .thelaw_1, C_2) : prop)]) =>
    ({def} Simp2

```

```

(hyp2_27 Iff1
C_26 Ui Sc
(Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1
(.M_1, .thelaw_1, C_26)]], [(hyp1_27
: that thetachain1
(.M_1, .thelaw_1, C_26)) =>
({def} (C_26
E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) Fixform
Simp1 (Simp2
(hyp1_27)) Iff2
C_26 Ui Scthm
(Sc (.M_1)) Conj
hyp1_27 Iff2
C_26 Ui Sc
(Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)] : that
C_26 E Sc (Sc
(.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)])]] : that
(C_26 E Sc (Sc
(.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_26)]) Mp
(Sc (Sc (.M_1)) Set
[(C_2 : obj) =>

```

```

      ({def} thetachain1
      (.M_1, .thelaw_1, C_2) : prop)]) Intax
    Sc (.M_1) : that
      J_5 E K_8]]))) Conj
Simp2 (thehyp_6) Mp
J_5 Ui H_4 Ui Simp2
(Simp2 (Simp2 (ktheta_9
Iff1 K_8 Ui Ug ([C_18
: obj) =>
({def} Dediff ([hyp2_19
: that C_18 E Sc
(Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_2) : prop)]) =>
({def} Simp2
(hyp2_19 Iff1
C_18 Ui Sc (Sc
(.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1 (.M_1, .thelaw_1, C_18))), [(hyp1_19
: that thetachain1
(.M_1, .thelaw_1, C_18)) =>
({def} (C_18
E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
  (.M_1, .thelaw_1, C_2) : prop)]) Fixform
Simp1 (Simp2
(hyp1_19)) Iff2
C_18 Ui Scthm
(Sc (.M_1)) Conj
hyp1_19 Iff2 C_18
Ui Sc (Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1

```

```

      (.M_1, .thelaw_1, C_2) : prop))] : that
C_18 E Sc (Sc
(.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))]] : that
(C_18 E Sc (Sc
(.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))] ==
  thetachain1 (.M_1, .thelaw_1, C_18))]])) : that
(H_4 Intersection J_5) E K_8))] : that
(K_8 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))] ->
(H_4 Intersection J_5) E K_8))] Iff2
(H_4 Intersection J_5) Ui
Forall ([(C_10 : obj) =>
  ({def} (C_10 E Misset_1
Mbold2 thelawchooses_1) ==
Forall ([(D_12 : obj) =>
  ({def} (D_12 E Sc
  (Sc (.M_1)) Set
  [(C_2 : obj) =>
    ({def} thetachain1
      (.M_1, .thelaw_1, C_2) : prop))] ->
    C_10 E D_12 : prop))] : prop))] Fixform
Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug ([(C_13
: obj) =>
  ({def} Dediff ([(hyp2_14
: that C_13 E Sc (Sc
(.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))] =>

```



```

({def} Simp2 (hyp2_14
Iff1 C_13 Ui Sc (Sc
(.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1 (.M_1, .thelaw_1, C_13))), [(hyp1_14
: that thetachain1 (.M_1, .thelaw_1, C_13)) =>
({def} (C_13 E Sc
(Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) Fixform
Simp1 (Simp2 (hyp1_14)) Iff2
C_13 Ui Scthm (Sc (.M_1)) Conj
hyp1_14 Iff2 C_13 Ui
Sc (Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)] : that
C_13 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)))]]) : that
(C_13 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_13)))] Mp
(Sc (Sc (.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) Intax
Sc (.M_1) : that (H_4 Intersection
J_5) E Misset_1 Mbold2 thelawchooses_1)]) : that
((H_4 <=< Misset_1 Mbold2 thelawchooses_1) & J_5
E H_4) -> (H_4 Intersection
J_5) E Misset_1 Mbold2 thelawchooses_1)]) : that
Forall [(x'_5 : obj) =>

```

```

      ({def} ((H_4 <= Misset_1
Mbold2 thelawchooses_1) & x'_5
E H_4) -> (H_4 Intersection
x'_5) E Misset_1 Mbold2 thelawchooses_1
: prop)))) : that Forall
([ (x'_4 : obj) =>
  ({def} Forall ([ (x'_5 : obj) =>
    ({def} ((x'_4 <= Misset_1
Mbold2 thelawchooses_1) & x'_5
E x'_4) -> (x'_4 Intersection
x'_5) E Misset_1 Mbold2 thelawchooses_1
: prop])) : prop)))]

```

```

Lineb14 : [(M_1 : obj), (Misset_1
: that Isset (M_1)), (thelaw_1
: [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
: [(S_2 : obj), (subtevev_2 : that
.S_2 <= M_1), (inev_2 : that
Exists ([ (x_4 : obj) =>
  ({def} x_4 E .S_2 : prop])) =>
  (--- : that .thelaw_1 (.S_2) E .S_2))] =>
(--- : that Forall ([ (x'_4 : obj) =>
  ({def} Forall ([ (x'_5 : obj) =>
    ({def} ((x'_4 <= Misset_1
Mbold2 thelawchooses_1) & x'_5
E x'_4) -> (x'_4 Intersection
x'_5) E Misset_1 Mbold2 thelawchooses_1
: prop])) : prop)))]

```

{move 0}

>>> open

{move 2}

```
>>> define Line14 : Lineb14 Misset, thelawchooses
```

```
Line14 : [
  ({def} Misset Lineb14 thelawchooses
  : that Forall ([x'_2 : obj) =>
    ({def} Forall ([x'_3 : obj) =>
      ({def} ((x'_2 <= Misset
        Mbold2 thelawchooses) & x'_3
        E x'_2) -> (x'_2 Intersection
        x'_3) E Misset Mbold2 thelawchooses
        : prop))) : prop]]])
```

```
Line14 : that Forall ([x'_2 : obj) =>
  ({def} Forall ([x'_3 : obj) =>
    ({def} ((x'_2 <= Misset Mbold2
    thelawchooses) & x'_3 E x'_2) ->
    (x'_2 Intersection x'_3) E Misset
    Mbold2 thelawchooses : prop])) : prop)])
```

```
{move 1}
end Lestrade execution
```

Here is the fourth component of the proof that **M** is a Θ -chain.

```
begin Lestrade execution
```

```
>>> define Mboldtheta1 : Fixform (thetachain \
  (Mbold), Conj (Line1, Conj (Line4, Conj \
  (Line8, Line14))))
```

```
Mboldtheta1 : [
  ({def} thetachain (Mbold) Fixform
  Line1 Conj Line4 Conj Line8 Conj
```

```

Line14 : that thetachain (Mbold))]

Mboldtheta1 : that thetachain (Mbold)

{move 1}

>>> close

{move 1}

>>> define Mboldtheta2 Misset thelawchooses \
      : Mboldtheta1

Mboldtheta2 : [(M_1 : obj), (Misset_1
      : that Isset (M_1)), (.thelaw_1
      : [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
      : [(S_2 : obj), (subsevev_2 : that
      .S_2 <= .M_1), (inev_2 : that
      Exists [(x_4 : obj) =>
      ({def} x_4 E .S_2 : prop)])]) =>
      (--- : that .thelaw_1 (.S_2) E .S_2)]) =>
      ({def} thetachain1 (M_1, .thelaw_1, Misset_1
      Mbold2 thelawchooses_1) Fixform Ug
      [(F_7 : obj) =>
      ({def} Ded [(ftheta_8 : that
      F_7 E Sc (Sc (M_1)) Set
      [(C_2 : obj) =>
      ({def} thetachain1 (M_1, .thelaw_1, C_2) : prop)]) =>
      ({def} Simp1 (ftheta_8 Iff1
      F_7 Ui Ug [(C_12 : obj) =>
      ({def} Dediff [(hyp2_13
      : that C_12 E Sc (Sc (M_1)) Set
      [(C_2 : obj) =>
      ({def} thetachain1

```

```

      (.M_1, .thelaw_1, C_2) : prop))] =>
({def} Simp2 (hyp2_13
Iff1 C_12 Ui Sc (Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))] : that
thetachain1 (.M_1, .thelaw_1, C_12))] , [(hyp1_13
: that thetachain1 (.M_1, .thelaw_1, C_12)) =>
({def} (C_12 E Sc (Sc
(.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))] Fixform
Simp1 (Simp2 (hyp1_13)) Iff2
C_12 Ui Scthm (Sc (.M_1)) Conj
hyp1_13 Iff2 C_12 Ui Sc
(Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop)] : that
C_12 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
    (.M_1, .thelaw_1, C_2) : prop))]] : that
(C_12 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ==
thetachain1 (.M_1, .thelaw_1, C_12))]] : that
.M_1 E F_7]] : that (F_7
E Sc (Sc (.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ->
.M_1 E F_7]] Iff2 .M_1 Ui Forall
([(C_9 : obj) =>
  ({def} (C_9 E Misset_1 Mbold2
thelawchooses_1) == Forall [(D_11
: obj) =>
  ({def} (D_11 E Sc (Sc (.M_1)) Set

```

```

      [(C_2 : obj) =>
        ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ->
      C_9 E D_11 : prop))) : prop))) Fixform
Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug [(C_12 : obj) =>
  ({def} Dediff [(hyp2_13 : that
    C_12 E Sc (Sc (.M_1)) Set
    [(C_2 : obj) =>
      ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] =>
      ({def} Simp2 (hyp2_13 Iff1
        C_12 Ui Sc (Sc (.M_1)) Separation
        [(C_2 : obj) =>
          ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] : that
          thetachain1 (.M_1, .thelaw_1, C_12))], [(hyp1_13
            : that thetachain1 (.M_1, .thelaw_1, C_12)) =>
            ({def} (C_12 E Sc (Sc (.M_1)) Set
              [(C_2 : obj) =>
                ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] Fixform
                Simp1 (Simp2 (hyp1_13)) Iff2
                C_12 Ui Scthm (Sc (.M_1)) Conj
                hyp1_13 Iff2 C_12 Ui Sc (Sc
                  (.M_1)) Separation [(C_2
                    : obj) =>
                      ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)] : that
                      C_12 E Sc (Sc (.M_1)) Set
                      [(C_2 : obj) =>
                        ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))]]] : that
                      (C_12 E Sc (Sc (.M_1)) Set
                      [(C_2 : obj) =>
                        ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ==
                        thetachain1 (.M_1, .thelaw_1, C_12)))] Mp
                      (Sc (Sc (.M_1)) Set [(C_2 : obj) =>
                        ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] Intax
                      Sc (.M_1) Conj ((Misset_1 Mbold2
                      thelawchooses_1 <=< Sc (.M_1)) Fixform
                      Ug [(A_9 : obj) =>
                        ({def} Ded [(ainev_10 : that
                          A_9 E Misset_1 Mbold2 thelawchooses_1) =>

```

```

({def} Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug ([C_14
: obj) =>
({def} Dediff ([hyp2_15
: that C_14 E Sc (Sc (.M_1)) Set
[C_2 : obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) =>
({def} Simp2 (hyp2_15
Iff1 C_14 Ui Sc (Sc (.M_1)) Separation
[C_2 : obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1 (.M_1, .thelaw_1, C_14))] , [(hyp1_15
: that thetachain1 (.M_1, .thelaw_1, C_14)) =>
({def} (C_14 E Sc (Sc
(.M_1)) Set [C_2
: obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]) Fixform
Simp1 (Simp2 (hyp1_15)) Iff2
C_14 Ui Scthm (Sc (.M_1)) Conj
hyp1_15 Iff2 C_14 Ui Sc
(Sc (.M_1)) Separation
[C_2 : obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)] : that
C_14 E Sc (Sc (.M_1)) Set
[C_2 : obj) =>
({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop)]]) : that
(C_14 E Sc (Sc (.M_1)) Set
[C_2 : obj) =>
({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) ==
thetachain1 (.M_1, .thelaw_1, C_14))] Mp
Sc (.M_1) Ui ainev_10 Iff1
A_9 Ui Forall ([C_16 : obj) =>
({def} (C_16 E Misset_1

```

```

Mbold2 thelawchooses_1) ==
Forall ([ (D_18 : obj) =>
  ({def} (D_18 E Sc (Sc
    (.M_1)) Set [(C_2
      : obj) =>
        ({def} thetachain1
          (.M_1, .thelaw_1, C_2) : prop)]) ->
          C_16 E D_18 : prop)]) : prop)]) Fixform
Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug ([ (C_19
  : obj) =>
    ({def} Dediff ([ (hyp2_20
      : that C_19 E Sc (Sc (.M_1)) Set
        [(C_2 : obj) =>
          ({def} thetachain1
            (.M_1, .thelaw_1, C_2) : prop)]) =>
          ({def} Simp2 (hyp2_20
            Iff1 C_19 Ui Sc (Sc (.M_1)) Separation
              [(C_2 : obj) =>
                ({def} thetachain1
                  (.M_1, .thelaw_1, C_2) : prop)]) : that
                thetachain1 (.M_1, .thelaw_1, C_19))], [(hyp1_20
                  : that thetachain1 (.M_1, .thelaw_1, C_19)) =>
                  ({def} (C_19 E Sc (Sc
                    (.M_1)) Set [(C_2
                      : obj) =>
                        ({def} thetachain1
                          (.M_1, .thelaw_1, C_2) : prop)]) Fixform
                    Simp1 (Simp2 (hyp1_20)) Iff2
                    C_19 Ui Scthm (Sc (.M_1)) Conj
                    hyp1_20 Iff2 C_19 Ui Sc
                    (Sc (.M_1)) Separation
                      [(C_2 : obj) =>
                        ({def} thetachain1
                          (.M_1, .thelaw_1, C_2) : prop)] : that
                    C_19 E Sc (Sc (.M_1)) Set
                      [(C_2 : obj) =>
                        ({def} thetachain1

```



```

      (.M_1, .thelaw_1, C_2) : prop]])) : that
(C_19 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ==
thetachain1 (.M_1, .thelaw_1, C_19))) Mp
(Sc (Sc (.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] Intax
Sc (.M_1) : that A_9 E Sc (.M_1))]) : that
(A_9 E Misset_1 Mbold2 thelawchooses_1) ->
A_9 E Sc (.M_1))]) Conj Inhabited
(Ug [(F_12 : obj) =>
  ({def} Ded [(ftheta_13 : that
F_12 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] =>
  ({def} Simp1 (ftheta_13 Iff1
F_12 Ui Ug [(C_17 : obj) =>
  ({def} Dediff [(hyp2_18
: that C_17 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop))] =>
  ({def} Simp2 (hyp2_18
Iff1 C_17 Ui Sc (Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop))] : that
thetachain1 (.M_1, .thelaw_1, C_17))], [(hyp1_18
: that thetachain1 (.M_1, .thelaw_1, C_17)) =>
  ({def} (C_17 E Sc (Sc
(.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1
(.M_1, .thelaw_1, C_2) : prop))] Fixform
Simp1 (Simp2 (hyp1_18)) Iff2
C_17 Ui Scthm (Sc (.M_1)) Conj
hyp1_18 Iff2 C_17 Ui Sc

```

```

      (Sc (.M_1)) Separation
      [(C_2 : obj) =>
        ({def} thetachain1
          (.M_1, .thelaw_1, C_2) : prop))] : that
      C_17 E Sc (Sc (.M_1)) Set
      [(C_2 : obj) =>
        ({def} thetachain1
          (.M_1, .thelaw_1, C_2) : prop))]] : that
      (C_17 E Sc (Sc (.M_1)) Set
        [(C_2 : obj) =>
          ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) ==
          thetachain1 (.M_1, .thelaw_1, C_17))]] : that
      .M_1 E F_12))] : that (F_12
E Sc (Sc (.M_1)) Set [(C_2
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) ->
.M_1 E F_12))] Iff2 .M_1 Ui Forall
([ (C_14 : obj) =>
  ({def} (C_14 E Misset_1 Mbold2
thelawchooses_1) == Forall ([ (D_16
: obj) =>
  ({def} (D_16 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) ->
C_14 E D_16 : prop))] : prop)]) Fixform
Misset_1 thetascm2 thelawchooses_1
Iff2 Sc (.M_1) Ui Ug ([ (C_17 : obj) =>
  ({def} Dediff ([ (hyp2_18 : that
C_17 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) =>
  ({def} Simp2 (hyp2_18 Iff1
C_17 Ui Sc (Sc (.M_1)) Separation
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) : that
thetachain1 (.M_1, .thelaw_1, C_17))], [(hyp1_18
: that thetachain1 (.M_1, .thelaw_1, C_17)) =>
  ({def} (C_17 E Sc (Sc (.M_1)) Set

```

```

[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] Fixform
Simp1 (Simp2 (hyp1_18)) Iff2
C_17 Ui Scthm (Sc (.M_1)) Conj
hyp1_18 Iff2 C_17 Ui Sc (Sc
(.M_1)) Separation [(C_2
: obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)] : that
C_17 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)))] : that
(C_17 E Sc (Sc (.M_1)) Set
[(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop))] ==
thetachain1 (.M_1, .thelaw_1, C_17)))] Mp
(Sc (Sc (.M_1)) Set [(C_2 : obj) =>
  ({def} thetachain1 (.M_1, .thelaw_1, C_2) : prop)]) Intax
Sc (.M_1)) Conj Sc2 (.M_1)) Conj
Misset_1 Lineb8 thelawchooses_1 Conj
Misset_1 Lineb14 thelawchooses_1 : that
thetachain1 (.M_1, .thelaw_1, Misset_1
Mbold2 thelawchooses_1))]

```

```

Mboldtheta2 : [(M_1 : obj), (Misset_1
: that Isset (.M_1)), (.thelaw_1
: [(S_2 : obj) => (--- : obj)]), (thelawchooses_1
: [(S_2 : obj), (subsevev_2 : that
.S_2 <= .M_1), (inev_2 : that
Exists [(x_4 : obj) =>
  ({def} x_4 E .S_2 : prop)])]) =>
(--- : that .thelaw_1 (.S_2) E .S_2))] =>
(--- : that thetachain1 (.M_1, .thelaw_1, Misset_1
Mbold2 thelawchooses_1))]

```

{move 0}

```

>>> open

{move 2}

>>> define Mboldtheta : Mboldtheta2 \
      Misset, thelawchooses

Mboldtheta : [
  ({def} Misset Mboldtheta2 thelawchooses
   : that thetachain1 (M, [(S'_2
    : obj) =>
    ({def} thelaw (S'_2) : obj)], Misset
    Mbold2 thelawchooses))]

Mboldtheta : that thetachain1 (M, [(S'_2
  : obj) =>
  ({def} thelaw (S'_2) : obj)], Misset
  Mbold2 thelawchooses)

{move 1}
end Lestrade execution

Mboldtheta asserts that M is a  $\Theta$ -chain.

```