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

M. Randall Holmes

June 5, 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 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.

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 $\mathbf{M}$ 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.

```
Lestrade execution:



load whatismath2
clearcurrent



declare M obj
```

```
>> M: obj {move 1}


declare Misset that Isset M

>> Misset: that Isset(M) {move 1}


open

   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 inev that Exists [x => x E S] \



>>    inev: that Exists([(x_1:obj) => ((x_1
>>         E S):prop)])
>>      {move 2}



   postulate thelaw S : obj

>>    thelaw: [(S_1:obj) => (---:obj)]
>>      {move 1}



   postulate thelawchooses subsetev inev \
      : that (thelaw S) E S

>>    thelawchooses: [(.S_1:obj),(subsetev_1:
>>         that (.S_1 <<= M)),(inev_1:that Exists([(x_2:
>>            obj) => ((x_2 E .S_1):prop)]))
>>         => (---:that (thelaw(.S_1) E .S_1))]
>>      {move 1}



   close
```

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.

```
Lestrade execution:


open
```

```
   declare S obj

>>    S: obj {move 2}



   declare subsetev that S <<= M

>>    subsetev: that (S <<= M) {move 2}



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

>>    prime1: [(S_1:obj) => (---:obj)]
>>      {move 1}



   save

   close

declare S1 obj

>> S1: obj {move 1}



define prime2 thelaw, S1:prime1 S1

>> prime2: [(thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (S1_1:obj) => ((S1_1 Complement Usc(thelaw_1(S1_1))):
>>      obj)]
>>   {move 0}
```

```
open

   define prime S: prime2 thelaw, S

>>    prime: [(S_1:obj) => (---:obj)]
>>      {move 1}
```

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.

```
Lestrade execution:


   open

      declare u obj

>>       u: obj {move 3}



      open

         declare hyp1 that u E prime S

>>          hyp1: that (u E prime(S)) {move
>>            4}



         declare hyp2 that (u E S) & ~(u \
            = thelaw S)
```

```
>>           hyp2: that ((u E S) & ~((u = thelaw(S))))
>>             {move 4}



         define line1 hyp1: Iff1(hyp1,Ui(u, \
            Compax(S, 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)))
>>                => (---:that ~((u E Usc(thelaw(S)))))]
>>             {move 3}



         open

            declare hyp3 that u = thelaw \
               S

>>             hyp3: that (u = thelaw(S)) {move
>>               5}



            define line3: Inusc2(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)))
>>                   => (---:that (u E Usc(thelaw(S))))]
>>                {move 4}



             define line5 hyp3: line4 hyp3 \
                Mp line2 hyp1

>>              line5: [(hyp3_1:that (u = thelaw(S)))
>>                   => (---:that ??)]
>>                {move 4}



             close

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

>>           line6: [(hyp1_1:that (u E prime(S)))
>>                => (---:that ((u E S) & ~((u
>>                = thelaw(S)))))]
>>             {move 3}



         open

            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))))
>>                  => (---:that ??)]
>>               {move 4}



            close

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

>>          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))))) => (---:
>>               that (u E (S Complement Usc(thelaw(S)))))]
>>            {move 3}



         close

      define bothways u : Dediff line6, line10


>>       bothways: [(u_1:obj) => (---:that ((u_1
>>           E prime(S)) == ((u_1 E S) & ~((u_1
>>           = thelaw(S))))))]
>>         {move 2}



      close

   define primeax subsetev : Ug bothways


>>    primeax: [(.S_1:obj),(subsetev_1:that
>>        (.S_1 <<= M)) => (---:that Forall([(u_12:
>>           obj) => (((u_12 E prime(.S_1)) ==
>>           ((u_12 E .S_1) & ~((u_12 = thelaw(.S_1))))):
>>           prop)]))
>>        ]
>>      {move 1}
```

```
   close
```

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.

```
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_1:obj) => (((M_1 E C11_1) & ((C11_1
>>      <<= Sc(M_1)) & (Forall([(D11_3:obj) =>
>>         (((D11_3 E C11_1) -> (prime2(thelaw_1,
>>         D11_3) E C11_1)):prop)])
>>      & Forall([(D11_4:obj) => (Forall([(F11_5:
>>            obj) => ((((D11_4 <<= C11_1) & (F11_5
>>            E D11_4)) -> ((D11_4 Intersection
>>            F11_5) E C11_1)):prop)])
>>         :prop)]))
>>      )):prop)]
>>   {move 0}



open

   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) => (---: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)) => (---:that Isset(.G_1))]
>>      {move 1}



   save
   close

declare C100 obj

>> C100: obj {move 1}



declare thetaev100 that thetachain C100

>> thetaev100: that thetachain(C100) {move 1}



declare G100 obj

>> G100: obj {move 1}



declare ginev100 that G100 E C100

>> ginev100: that (G100 E C100) {move 1}



define Setsinchains2 Misset thelawchooses, \
   thetaev100 ginev100: Setsinchains1 thetaev100 \
   ginev100

>> Setsinchains2: [(.M_1:obj),(Misset_1:that
>>      Isset(.M_1)),(.thelaw_1:[(S_2:obj) =>
>>         (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))]),
>>      (.C100_1:obj),(thetaev100_1:that thetachain1(.M_1,
>>      .thelaw_1,.C100_1)),(.G100_1:obj),(ginev100_1:
>>      that (.G100_1 E .C100_1)) => (Simp1(Simp2(((ginev100_1
>>      Mp (.G100_1 Ui Simp1(Simp1(Simp2(thetaev100_1)))))
>>      Iff1 (.G100_1 Ui Scthm(.M_1))))):that
>>      Isset(.G100_1))]
>>   {move 0}



open

   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)) => (---:that Isset(.G_1))]
>>      {move 1}



   close
```

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.

```
Lestrade execution:



open

   open

      define line1 : Misset Mp Ui M Inownpowerset \
>>        line1: [(---:that (M E Sc(M)))]
>>          {move 2}



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

>>        line2: [(---:that (Sc(M) <<= Sc(M)))]
>>          {move 2}
```

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

```
Lestrade execution:



      open

         declare u obj

>>          u: obj {move 4}




         open

            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: [(---:that Isset(prime(u)))]
>>               {move 4}
```

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.

```
Lestrade execution:


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

>>             line4: [(usinev_1:that (u E Sc(M)))
>>                  => (---:that Isset(u))]
>>               {move 4}



            open

               declare v obj

>>                v: obj {move 6}



               open
               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: [(---: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))) => (---:
>>                     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))) => (---:
>>                     that (v E M))]
>>                  {move 6}
```

```
                  close

               define line8 v: Ded line7


>>                line8: [(v_1:obj) => (---:
>>                     that ((v_1 E prime(u))
>>                     -> (v_1 E M)))]
>>                  {move 5}



               close

            define line9 usinev: Ug line8


>>             line9: [(usinev_1:that (u E Sc(M)))
>>                  => (---:that Forall([(v_10:
>>                     obj) => (((v_10 E prime(u))
>>                     -> (v_10 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))) => (---: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))) => (---:that (prime(u)
>>                    E Sc(M)))]
>>                 {move 4}



              close

           define line12 u: Ded line11

>>            line12: [(u_1:obj) => (---:that
>>                 ((u_1 E Sc(M)) -> (prime(u_1)
>>                 E Sc(M))))]
>>              {move 3}



           close

        define line13 : Ug line12

>>         line13: [(---:that Forall([(u_14:obj)
>>                 => (((u_14 E Sc(M)) -> (prime(u_14)
>>                 E Sc(M))):prop)]))
>>              ]
>>           {move 2}
```

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

```
Lestrade execution:
      open

         declare u obj

>>          u: obj {move 4}



         open

            declare v obj

>>             v: obj {move 5}



            open

               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))) => (---:
>>                     that Forall([(u_4:obj)
>>                        => (((u_4 E (u Intersection
>>                        v)) == Forall([(B1_5:
>>                           obj) => (((B1_5 E
>>                           u) -> (u_4 E B1_5)):
```

```
>>                           prop)]))
>>                        :prop)]))
>>                     ]
>>                  {move 5}



             open

                declare w obj

>>                 w: obj {move 7}



                open

                   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))) => (---:that
>>                         (w E v))]
>>                      {move 7}
```

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

>>                   line16: [(---:that (v
>>                        E Sc(M)))]
>>                     {move 7}



                  define line17 : Iff1(line16, \
                     Ui v Scthm 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))) => (---:that
>>                        (w E M))]
>>                     {move 7}



                  close

               define line19 w: Ded line18


>>                line19: [(w_1:obj) => (---:
```

```
>>                          that ((w_1 E (u Intersection
>>                          v)) -> (w_1 E M)))]
>>                       {move 6}



                   close

                define line20 hyp : Ug line19


>>                 line20: [(hyp_1:that ((u <<=
>>                      Sc(M)) & (v E u))) => (---:
>>                      that Forall([(w_14:obj)
>>                         => (((w_14 E (u Intersection
>>                         v)) -> (w_14 E M)):prop)]))
>>                      ]
>>                   {move 5}



                define line21 : Fixform(Isset(Intersection \
                   u v),Separation3 (Refleq(Intersection \
                   u 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))) => (---:
>>                      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))) => (---:
>>                     that ((u Intersection v)
>>                     E Sc(M)))]
>>                  {move 5}



               close

            define line24 v: Ded line23

>>             line24: [(v_1:obj) => (---:that
>>                  (((u <<= Sc(M)) & (v_1 E u))
>>                  -> ((u Intersection v_1) E
>>                  Sc(M))))]
>>               {move 4}



            close

         define line25 u: Ug line24

>>          line25: [(u_1:obj) => (---:that
>>               Forall([(v_23:obj) => ((((u_1
>>                  <<= Sc(M)) & (v_23 E u_1))
>>                  -> ((u_1 Intersection v_23)
>>                  E Sc(M))):prop)]))
>>               ]
>>            {move 3}



         close

      define line26: Ug line25

>>       line26: [(---:that Forall([(u_24:obj)
>>               => (Forall([(v_25:obj) => ((((u_24
>>                  <<= Sc(M)) & (v_25 E u_24))
>>                  -> ((u_24 Intersection v_25)
>>                  E Sc(M))):prop)])
>>               :prop)]))
>>            ]
>>         {move 2}
```

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

```
Lestrade execution:



      close

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

>>    thetascm1: [(---:that thetachain(Sc(M)))]
>>      {move 1}



   close
```

```
define thetascm2 Misset thelawchooses: thetascm1


>> thetascm2: [(.M_1:obj),(Misset_1:that Isset(.M_1)),
>>      (.thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))])
>>      => ((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_16:
>>         obj) => (Ded([(usinev_17:that (u_16
>>            E Sc(.M_1))) => ((((prime2(.thelaw_1,
>>            u_16) <<= .M_1) Fixform (Ug([(v_20:
>>               obj) => (Ded([(vinev_21:that
>>                  (v_20 E prime2(.thelaw_1,u_16)))
>>                  => ((Simp1((vinev_21 Iff1
>>                  (v_20 Ui Ug([(u_24:obj) =>
>>                     (Dediff([(hyp1_25:that
>>                        (u_24 E prime2(.thelaw_1,
>>                        u_16))) => ((Simp1((hyp1_25
>>                        Iff1 (u_24 Ui (u_16
>>                        Compax Usc(.thelaw_1(u_16))))))
>>                        Conj Negintro([(hyp3_27:
>>                           that (u_24 = .thelaw_1(u_16)))
>>                           => ((Subs(Eqsymm(hyp3_27),
>>                           [(v_28:obj) => ((v_28
>>                              E Usc(.thelaw_1(u_16))):
>>                              prop)]
>>                           ,Inusc2(.thelaw_1(u_16)))
>>                           Mp Simp2((hyp1_25
>>                           Iff1 (u_24 Ui (u_16
>>                           Compax Usc(.thelaw_1(u_16))))))):
>>                           that ??)]))
>>                       :that ((u_24 E u_16)
>>                       & ~((u_24 = .thelaw_1(u_16)))))]
>>                    ,[(hyp2_30:that ((u_24
>>                       E u_16) & ~((u_24 =
>>                       .thelaw_1(u_16)))))
>>                       => (((Simp1(hyp2_30)
>>                       Conj Negintro([(hyp4_31:
>>                          that (u_24 E Usc(.thelaw_1(u_16))))
>>                          => ((Inusc1(hyp4_31)
>>                          Mp Simp2(hyp2_30)):
>>                          that ??)]))
>>                       Iff2 (u_24 Ui (u_16
>>                       Compax Usc(.thelaw_1(u_16))))):
>>                       that (u_24 E (u_16 Complement
>>                       Usc(.thelaw_1(u_16)))))])
>>                    :that ((u_24 E prime2(.thelaw_1,
>>                    u_16)) == ((u_24 E u_16)
>>                    & ~((u_24 = .thelaw_1(u_16))))))]))
>>                 )) Mp (v_20 Ui Simp1((usinev_17
>>                 Iff1 (u_16 Ui Scthm(.M_1)))))):
>>                 that (v_20 E .M_1))])
>>              :that ((v_20 E prime2(.thelaw_1,
>>              u_16)) -> (v_20 E .M_1)))])
>>           Conj ((Isset(prime2(.thelaw_1,u_16))
>>           Fixform Separation3(Refleq(prime2(.thelaw_1,
>>           u_16)))) Conj Misset_1))) Iff2 (prime2(.thelaw_1,
>>           u_16) Ui Scthm(.M_1))):that (prime2(.thelaw_1,
>>           u_16) E Sc(.M_1)))])
>>        :that ((u_16 E Sc(.M_1)) -> (prime2(.thelaw_1,
>>        u_16) E Sc(.M_1))))])
>>     Conj Ug([(u_42:obj) => (Ug([(v_44:obj)
>>           => (Ded([(hyp_45:that ((u_42 <<=
>>              Sc(.M_1)) & (v_44 E u_42))) =>
>>              (((((u_42 Intersection v_44)
>>              <<= .M_1) Fixform (Ug([(w_48:
>>                 obj) => (Ded([(hyp2_49:that
>>                    (w_48 E (u_42 Intersection
>>                    v_44))) => (((Simp2(hyp_45)
>>                      Mp (v_44 Ui (hyp2_49 Iff1
>>                      (w_48 Ui (Simp2(hyp_45)
>>                      Mp (u_42 Intax v_44))))))
>>                      Mp (w_48 Ui Simp1(((Simp2(hyp_45)
>>                      Mp (v_44 Ui Simp1(Simp1(hyp_45))))
>>                      Iff1 (v_44 Ui Scthm(.M_1)))))):
>>                      that (w_48 E .M_1))])
>>                   :that ((w_48 E (u_42 Intersection
>>                   v_44)) -> (w_48 E .M_1)))])
>>                Conj ((Isset((u_42 Intersection
>>                v_44)) Fixform Separation3(Refleq((u_42
>>                Intersection v_44)))) Conj Misset_1)))
>>                Iff2 ((u_42 Intersection v_44)
>>                Ui Scthm(.M_1))):that ((u_42
>>                Intersection v_44) E Sc(.M_1)))])
>>             :that (((u_42 <<= Sc(.M_1)) & (v_44
>>             E u_42)) -> ((u_42 Intersection
>>             v_44) E Sc(.M_1))))])
>>          :that Forall([(v_64:obj) => ((((u_42
>>             <<= Sc(.M_1)) & (v_64 E u_42)) ->
>>             ((u_42 Intersection v_64) E Sc(.M_1))):
>>             prop)]))
>>          ]))
>>       ))):that thetachain1(.M_1,.thelaw_1,Sc(.M_1)))]
>>    {move 0}
```

```
open

   define thetascm: thetascm2 Misset thelawchooses
```

```
>>    thetascm: [(---:that thetachain1(M,thelaw,
>>         Sc(M)))]
>>      {move 1}
```

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.

```
Lestrade execution:

     clearcurrent


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


>>    Thetachain: [(---:obj)]
>>      {move 1}




   open

      declare C obj

>>       C: obj {move 3}



      open

         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))
>>               => (---: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))
>>               => (---: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))
>>               => (---:that thetachain(C))]
>>            {move 3}



         close

      define line4 C: Dediff line3, line2
```

```
>>       line4: [(C_1:obj) => (---:that ((C_1
>>            E Thetachain) == thetachain(C_1)))]
>>         {move 2}



      close

   define Thetachainax: Ug line4

>>    Thetachainax: [(---:that Forall([(C_13:
>>            obj) => (((C_13 E Thetachain) ==
>>            thetachain(C_13)):prop)]))
>>         ]
>>      {move 1}
```

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).

```
Lestrade execution:


   define Mbold1 : Intersection Thetachain \
      Sc M

>>    Mbold1: [(---:obj)]
>>      {move 1}



   close
```

```
define Mbold2 Misset thelawchooses:Mbold1


>> Mbold2: [(.M_1:obj),(Misset_1:that Isset(.M_1)),
>>      (.thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))])
>>      => (((Sc(Sc(.M_1)) Set [(C_5:obj) => (thetachain1(.M_1,
>>         .thelaw_1,C_5):prop)])
>>      Intersection Sc(.M_1)):obj)]
>>   {move 0}
```

```
open

   define Mbold: Mbold2 Misset thelawchooses


>>    Mbold: [(---:obj)]
>>      {move 1}
```

We now have the tedious task of directly verifying that **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 **M** itself is a Θ-chain.

```
Lestrade execution:

     clearcurrent
```

```
   declare C obj

>>    C: obj {move 2}



   declare D obj

>>    D: obj {move 2}



   open

      define Mboldax1: Intax Thetachain Sc \
         M

>>       Mboldax1: [(---:that ((Sc(M) E Thetachain)
>>            -> Forall([(u_1:obj) => (((u_1 E
>>               (Thetachain Intersection Sc(M)))
>>               == Forall([(B1_2:obj) => (((B1_2
>>                  E Thetachain) -> (u_1 E B1_2)):
>>                  prop)]))
>>               :prop)]))
>>            )]
>>         {move 2}



      define line1: Ui Sc M Thetachainax


>>       line1: [(---:that ((Sc(M) E Thetachain)
>>            == thetachain(Sc(M))))]
>>         {move 2}
```

```
      define line2: Iff2 thetascm line1

>>       line2: [(---:that (Sc(M) E Thetachain))]
>>         {move 2}



      close

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

>>    Mboldax: [(---:that Forall([(C_5:obj)
>>          => (((C_5 E Mbold) == Forall([(D_6:
>>             obj) => (((D_6 E Thetachain)
>>             -> (C_5 E D_6)):prop)]))
>>          :prop)]))
>>       ]
>>     {move 1}
```

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.

```
Lestrade execution:

     clearcurrent


   open
```

```
      declare F obj

>>       F: obj {move 3}



      open

         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))
>>               => (---:that thetachain(F))]
>>            {move 3}




         define line2 ftheta: Simp1 line1 \
            ftheta

>>          line2: [(ftheta_1:that (F E Thetachain))
>>               => (---:that (M E F))]
>>            {move 3}




         close

      define Linea1 F: Ded line2
```

```
>>        Linea1: [(F_1:obj) => (---:that ((F_1
>>             E Thetachain) -> (M E F_1)))]
>>          {move 2}



      close

   define Lineb1: Ug Linea1

>>    Lineb1: [(---:that Forall([(F_7:obj) =>
>>            (((F_7 E Thetachain) -> (M E F_7)):
>>            prop)]))
>>          ]
>>      {move 1}



   define Line1 : Iff2(Lineb1,Ui M Mboldax)


>>    Line1: [(---:that (M E Mbold))]
>>      {move 1}


          clearcurrent
```

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

```
Lestrade execution:



   open

      open

         declare A obj
```

```
>>          A: obj {move 4}



         open

            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))
>>                  => (---:that (A E Sc(M)))]
>>               {move 4}



            close

         define line2 A: Ded line1

>>          line2: [(A_1:obj) => (---:that ((A_1
>>               E Mbold) -> (A_1 E Sc(M))))]
>>            {move 3}



         close
```

```
      define Line3 : Ug line2

>>       Line3: [(---:that Forall([(A_8:obj)
>>               => (((A_8 E Mbold) -> (A_8 E
>>               Sc(M))):prop)]))
>>            ]
>>         {move 2}



      close

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

>>    Line4: [(---:that (Mbold <<= Sc(M)))]
>>      {move 1}


          clearcurrent
```

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

```
Lestrade execution:


   open

      declare F obj

>>       F: obj {move 3}



      open
```

```
          declare finmbold that F E (Mbold)


>>           finmbold: that (F E Mbold) {move
>>             4}



          open

             declare G obj

>>              G: obj {move 5}



             open

                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)) => (---:
>>                      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)) => (---:
>>                    that (F E G))]
>>                 {move 5}



              define line3 gtheta : Mp line2 \
                 gtheta line1 gtheta

>>               line3: [(gtheta_1:that (G
>>                    E Thetachain)) => (---:
>>                    that (prime2(thelaw,F)
>>                    E G))]
>>                 {move 5}



              close

            define line4 G : Ded line3

>>             line4: [(G_1:obj) => (---:that
>>                  ((G_1 E Thetachain) -> (prime2(thelaw,
>>                  F) E G_1)))]
>>               {move 4}



            close

          define line5 finmbold: Ug line4
```

```
>>          line5: [(finmbold_1:that (F E Mbold))
>>               => (---:that Forall([(G_20:obj)
>>                  => (((G_20 E Thetachain) ->
>>                  (prime2(thelaw,F) E G_20)):
>>                  prop)]))
>>               ]
>>            {move 3}



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


>>          line6: [(finmbold_1:that (F E Mbold))
>>               => (---:that (prime(F) E Mbold))]
>>            {move 3}




         close

      define line7 F: Ded line6

>>       line7: [(F_1:obj) => (---:that ((F_1
>>            E Mbold) -> (prime(F_1) E Mbold)))]
>>         {move 2}




      close

   define Linea8 : Ug line7

>>    Linea8: [(---:that Forall([(F_24:obj)
>>            => (((F_24 E Mbold) -> (prime(F_24)
>>            E Mbold)):prop)]))
>>         ]
>>      {move 1}



   save

   close

define Lineb8 Misset thelawchooses: Linea8


>> Lineb8: [(.M_1:obj),(Misset_1:that Isset(.M_1)),
>>      (.thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))])
>>      => (Ug([(F_6:obj) => (Ded([(finmbold_7:
>>            that (F_6 E (Misset_1 Mbold2 thelawchooses_1)))
>>            => ((Ug([(G_12:obj) => (Ded([(gtheta_14:
>>                  that (G_12 E (Sc(Sc(.M_1))
>>                  Set [(C_15:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_15):prop)]))
>>                  ) => (((gtheta_14 Mp (G_12
>>                  Ui (finmbold_7 Iff1 (F_6 Ui
>>                  (Forall([(C_24:obj) => (((C_24
>>                     E (Misset_1 Mbold2 thelawchooses_1))
>>                     == Forall([(D_25:obj) =>
>>                        (((D_25 E (Sc(Sc(.M_1))
>>                        Set [(C_26:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_26):prop)]))
>>                        -> (C_24 E D_25)):prop)]))
>>                     :prop)])
>>                  Fixform (((Misset_1 thetascm2
>>                  thelawchooses_1) Iff2 (Sc(.M_1)
>>                  Ui Ug([(C_33:obj) => (Dediff([(hyp2_35:
>>                        that (C_33 E (Sc(Sc(.M_1))
>>                        Set [(C_36:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_36):prop)]))
>>                        ) => (Simp2((hyp2_35
>>                        Iff1 (C_33 Ui (Sc(Sc(.M_1))
>>                        Separation [(C_40:obj)
>>                           => (thetachain1(.M_1,
>>                           .thelaw_1,C_40):prop)]))
>>                        )):that thetachain1(.M_1,
>>                        .thelaw_1,C_33))]
>>                     ,[(hyp1_41:that thetachain1(.M_1,
>>                        .thelaw_1,C_33)) =>
>>                        (((C_33 E (Sc(Sc(.M_1))
>>                        Set [(C_42:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_42):prop)]))
>>                        Fixform (((Simp1(Simp2(hyp1_41))
>>                        Iff2 (C_33 Ui Scthm(Sc(.M_1))))
>>                        Conj hyp1_41) Iff2 (C_33
>>                        Ui (Sc(Sc(.M_1)) Separation
>>                        [(C_53:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_53):prop)]))
>>                        )):that (C_33 E (Sc(Sc(.M_1))
>>                        Set [(C_54:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_54):prop)]))
>>                        )])
>>                     :that ((C_33 E (Sc(Sc(.M_1))
>>                     Set [(C_55:obj) => (thetachain1(.M_1,
>>                        .thelaw_1,C_55):prop)]))
>>                     == thetachain1(.M_1,.thelaw_1,
>>                     C_33)))]))
>>                  ) Mp ((Sc(Sc(.M_1)) Set [(C_60:
>>                     obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_60):prop)])
>>                  Intax Sc(.M_1)))))))) Mp (F_6
>>                  Ui Simp1(Simp2(Simp2((gtheta_14
>>                  Iff1 (G_12 Ui Ug([(C_76:obj)
>>                     => (Dediff([(hyp2_78:that
>>                        (C_76 E (Sc(Sc(.M_1))
>>                        Set [(C_79:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_79):prop)]))
>>                        ) => (Simp2((hyp2_78
>>                        Iff1 (C_76 Ui (Sc(Sc(.M_1))
>>                        Separation [(C_83:obj)
>>                           => (thetachain1(.M_1,
>>                           .thelaw_1,C_83):prop)]))
>>                        )):that thetachain1(.M_1,
>>                        .thelaw_1,C_76))]
>>                     ,[(hyp1_84:that thetachain1(.M_1,
>>                        .thelaw_1,C_76)) =>
>>                        (((C_76 E (Sc(Sc(.M_1))
>>                        Set [(C_85:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_85):prop)]))
>>                        Fixform (((Simp1(Simp2(hyp1_84))
>>                        Iff2 (C_76 Ui Scthm(Sc(.M_1))))
>>                        Conj hyp1_84) Iff2 (C_76
>>                        Ui (Sc(Sc(.M_1)) Separation
>>                        [(C_96:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_96):prop)]))
>>                        )):that (C_76 E (Sc(Sc(.M_1))
>>                        Set [(C_97:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_97):prop)]))
>>                        )])
>>                     :that ((C_76 E (Sc(Sc(.M_1))
>>                     Set [(C_98:obj) => (thetachain1(.M_1,
>>                        .thelaw_1,C_98):prop)]))
>>                     == thetachain1(.M_1,.thelaw_1,
>>                     C_76)))]))
>>                  )))))):that (prime2(.thelaw_1,
>>                  F_6) E G_12))])
>>               :that ((G_12 E (Sc(Sc(.M_1))
>>               Set [(C_99:obj) => (thetachain1(.M_1,
>>                  .thelaw_1,C_99):prop)]))
>>               -> (prime2(.thelaw_1,F_6) E G_12)))])
>>            Iff2 (prime2(.thelaw_1,F_6) Ui (Forall([(C_103:
>>               obj) => (((C_103 E (Misset_1
>>               Mbold2 thelawchooses_1)) == Forall([(D_104:
>>                  obj) => (((D_104 E (Sc(Sc(.M_1))
>>                  Set [(C_105:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_105):prop)]))
>>                  -> (C_103 E D_104)):prop)]))
>>               :prop)])
>>            Fixform (((Misset_1 thetascm2 thelawchooses_1)
>>            Iff2 (Sc(.M_1) Ui Ug([(C_112:obj)
>>               => (Dediff([(hyp2_114:that (C_112
>>                  E (Sc(Sc(.M_1)) Set [(C_115:
>>                     obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_115):prop)]))
>>                  ) => (Simp2((hyp2_114 Iff1
>>                  (C_112 Ui (Sc(Sc(.M_1)) Separation
>>                  [(C_119:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_119):prop)]))
>>                  )):that thetachain1(.M_1,.thelaw_1,
>>                  C_112))]
>>               ,[(hyp1_120:that thetachain1(.M_1,
>>                  .thelaw_1,C_112)) => (((C_112
>>                  E (Sc(Sc(.M_1)) Set [(C_121:
>>                     obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_121):prop)]))
>>                  Fixform (((Simp1(Simp2(hyp1_120))
>>                  Iff2 (C_112 Ui Scthm(Sc(.M_1))))
>>                  Conj hyp1_120) Iff2 (C_112
>>                  Ui (Sc(Sc(.M_1)) Separation
>>                  [(C_132:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_132):prop)]))
>>                  )):that (C_112 E (Sc(Sc(.M_1))
>>                  Set [(C_133:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_133):prop)]))
>>                  )])
>>               :that ((C_112 E (Sc(Sc(.M_1))
>>               Set [(C_134:obj) => (thetachain1(.M_1,
>>                  .thelaw_1,C_134):prop)]))
>>               == thetachain1(.M_1,.thelaw_1,
>>               C_112)))]))
>>            ) Mp ((Sc(Sc(.M_1)) Set [(C_139:
>>               obj) => (thetachain1(.M_1,.thelaw_1,
>>                C_139):prop)])
>>              Intax Sc(.M_1)))))):that (prime2(.thelaw_1,
>>              F_6) E (Misset_1 Mbold2 thelawchooses_1)))])
>>           :that ((F_6 E (Misset_1 Mbold2 thelawchooses_1))
>>           -> (prime2(.thelaw_1,F_6) E (Misset_1
>>           Mbold2 thelawchooses_1))))])
>>        :that Forall([(F_140:obj) => (((F_140
>>           E (Misset_1 Mbold2 thelawchooses_1))
>>           -> (prime2(.thelaw_1,F_140) E (Misset_1
>>           Mbold2 thelawchooses_1))):prop)]))
>>        ]
>>     {move 0}
```

```
open

   define Line8: Lineb8 Misset thelawchooses
```

```
>>      Line8: [(---:that Forall([(F_1:obj) =>
>>              (((F_1 E (Misset Mbold2 thelawchooses))
>>              -> (prime2(thelaw,F_1) E (Misset
>>              Mbold2 thelawchooses))):prop)]))
>>           ]
>>        {move 1}
```

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

```
Lestrade execution:


   open

      declare H obj
```

```
>>      H: obj {move 3}



      open

         declare J obj

>>          J: obj {move 4}



         open

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

>>             thehyp: that ((H <<= Mbold) &
>>               (J E H)) {move 5}



            open

               declare K obj

>>                K: obj {move 6}



               open

                  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)) =>
>>                          (---:that thetachain(K))]
>>                       {move 6}



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

>>                     line2: [(ktheta_1:that
>>                          (K E Thetachain)) =>
>>                          (---:that (((H <<= K)
>>                          & (J E H)) -> ((H Intersection
>>                          J) E K)))]
>>                       {move 6}



                    open

                       declare P obj

>>                        P: obj {move 8}



                       open

                          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)) => (---:
>>                                 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)) => (---:
>>                                 that (P E K))]
>>                              {move 8}



                           close

                        define line5 P :Ded \
                           line4

>>                         line5: [(P_1:obj) =>
>>                              (---:that ((P_1 E
>>                              H) -> (P_1 E K)))]
>>                           {move 7}
```

```
                   close

                define test1 ktheta: Ug \
                   line5

>>                 test1: [(ktheta_1:that
>>                      (K E Thetachain)) =>
>>                      (---:that Forall([(P_11:
>>                         obj) => (((P_11 E
>>                         H) -> (P_11 E K)):
>>                         prop)]))
>>                      ]
>>                   {move 6}



                define test2 ktheta: Inhabited \
                   Simp2 thehyp

>>                 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)) =>
>>                      (---: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)) =>
>>                        (---:that (H <<= K))]
>>                     {move 6}



                  define linea7 ktheta: Mp(Conj(line6 \
                     ktheta, Simp2 thehyp), \
                     line2 ktheta)

>>                   linea7: [(ktheta_1:that
>>                        (K E Thetachain)) =>
>>                        (---:that ((H Intersection
>>                        J) E K))]
>>                     {move 6}



                  close

               define line8 K: Ded linea7


>>                line8: [(K_1:obj) => (---:
>>                     that ((K_1 E Thetachain)
>>                     -> ((H Intersection J)
>>                     E K_1)))]
>>                  {move 5}
```

```
               close

            define line9 thehyp: Ug line8


>>             line9: [(thehyp_1:that ((H <<=
>>                  Mbold) & (J E H))) => (---:
>>                  that Forall([(K_35:obj) =>
>>                     (((K_35 E Thetachain) ->
>>                     ((H Intersection J) E K_35)):
>>                     prop)]))
>>                  ]
>>               {move 4}



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

>>             line10: [(---:that (((H Intersection
>>                  J) E Mbold) == Forall([(D_3:
>>                     obj) => (((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))) => (---:
>>                  that ((H Intersection J) E
>>                  Mbold))]
>>               {move 4}
```

```
            close

         define line12 J: Ded line11

>>          line12: [(J_1:obj) => (---:that
>>               (((H <<= Mbold) & (J_1 E H))
>>               -> ((H Intersection J_1) E Mbold)))]
>>            {move 3}



         close

      define line13 H: Ug line12

>>       line13: [(H_1:obj) => (---:that Forall([(J_41:
>>               obj) => ((((H_1 <<= Mbold) &
>>               (J_41 E H_1)) -> ((H_1 Intersection
>>               J_41) E Mbold)):prop)]))
>>            ]
>>         {move 2}



      close

   define Linea14: Ug line13

>>    Linea14: [(---:that Forall([(H_42:obj)
>>            => (Forall([(J_43:obj) => ((((H_42
>>               <<= Mbold) & (J_43 E H_42)) ->
>>               ((H_42 Intersection J_43) E Mbold)):
>>               prop)])
>>            :prop)]))
>>         ]
>>      {move 1}
```

```
   save

   close

define Lineb14 Misset thelawchooses:Linea14


>> Lineb14: [(.M_1:obj),(Misset_1:that Isset(.M_1)),
>>      (.thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))])
>>      => (Ug([(H_7:obj) => (Ug([(J_9:obj) =>
>>            (Ded([(thehyp_10:that ((H_7 <<=
>>               (Misset_1 Mbold2 thelawchooses_1))
>>               & (J_9 E H_7))) => ((Ug([(K_15:
>>                  obj) => (Ded([(ktheta_17:that
>>                     (K_15 E (Sc(Sc(.M_1)) Set
>>                     [(C_18:obj) => (thetachain1(.M_1,
>>                        .thelaw_1,C_18):prop)]))
>>                     ) => (((((H_7 <<= K_15)
>>                     Fixform (Ug([(P_21:obj)
>>                        => (Ded([(phyp_22:that
>>                           (P_21 E H_7)) =>
>>                           ((ktheta_17 Mp (K_15
>>                           Ui ((phyp_22 Mp (P_21
>>                           Ui Simp1(Simp1(thehyp_10))))
>>                           Iff1 (P_21 Ui (Forall([(C_33:
>>                              obj) => (((C_33
>>                              E (Misset_1 Mbold2
>>                              thelawchooses_1))
>>                              == Forall([(D_34:
>>                                 obj) => (((D_34
>>                                   E (Sc(Sc(.M_1))
>>                                   Set [(C_35:
>>                                      obj) =>
>>                                      (thetachain1(.M_1,
>>                                      .thelaw_1,
>>                                      C_35):prop)]))
>>                                   -> (C_33 E
>>                                   D_34)):prop)]))
>>                                :prop)])
>>                             Fixform (((Misset_1
>>                             thetascm2 thelawchooses_1)
>>                             Iff2 (Sc(.M_1) Ui
>>                             Ug([(C_42:obj) =>
>>                                (Dediff([(hyp2_44:
>>                                   that (C_42
>>                                   E (Sc(Sc(.M_1))
>>                                   Set [(C_45:
>>                                      obj) =>
>>                                      (thetachain1(.M_1,
>>                                      .thelaw_1,
>>                                      C_45):prop)]))
>>                                   ) => (Simp2((hyp2_44
>>                                   Iff1 (C_42
>>                                   Ui (Sc(Sc(.M_1))
>>                                   Separation
>>                                   [(C_49:obj)
>>                                      => (thetachain1(.M_1,
>>                                      .thelaw_1,
>>                                      C_49):prop)]))
>>                                   )):that thetachain1(.M_1,
>>                                   .thelaw_1,C_42))]
>>                                ,[(hyp1_50:that
>>                                   thetachain1(.M_1,
>>                                   .thelaw_1,C_42))
>>                                   => (((C_42
>>                                   E (Sc(Sc(.M_1))
>>                                   Set [(C_51:
>>                                      obj) =>
>>                                     (thetachain1(.M_1,
>>                                     .thelaw_1,
>>                                     C_51):prop)]))
>>                                  Fixform (((Simp1(Simp2(hyp1_50))
>>                                  Iff2 (C_42
>>                                  Ui Scthm(Sc(.M_1))))
>>                                  Conj hyp1_50)
>>                                  Iff2 (C_42
>>                                  Ui (Sc(Sc(.M_1))
>>                                  Separation
>>                                  [(C_62:obj)
>>                                     => (thetachain1(.M_1,
>>                                     .thelaw_1,
>>                                     C_62):prop)]))
>>                                  )):that (C_42
>>                                  E (Sc(Sc(.M_1))
>>                                  Set [(C_63:
>>                                     obj) =>
>>                                     (thetachain1(.M_1,
>>                                     .thelaw_1,
>>                                     C_63):prop)]))
>>                                  )])
>>                               :that ((C_42 E
>>                               (Sc(Sc(.M_1))
>>                               Set [(C_64:obj)
>>                                  => (thetachain1(.M_1,
>>                                  .thelaw_1,C_64):
>>                                  prop)]))
>>                               == thetachain1(.M_1,
>>                               .thelaw_1,C_42))]))
>>                            ) Mp ((Sc(Sc(.M_1))
>>                            Set [(C_69:obj) =>
>>                               (thetachain1(.M_1,
>>                               .thelaw_1,C_69):
>>                               prop)])
>>                            Intax Sc(.M_1))))))):
>>                            that (P_21 E K_15))])
>>                         :that ((P_21 E H_7)
>>                        -> (P_21 E K_15)))])
>>                     Conj (Inhabited(Simp2(thehyp_10))
>>                     Conj Inhabited((Simp2(thehyp_10)
>>                     Mp Ded([(phyp_70:that (J_9
>>                        E H_7)) => ((ktheta_17
>>                        Mp (K_15 Ui ((phyp_70
>>                        Mp (J_9 Ui Simp1(Simp1(thehyp_10))))
>>                        Iff1 (J_9 Ui (Forall([(C_81:
>>                           obj) => (((C_81 E
>>                           (Misset_1 Mbold2
>>                           thelawchooses_1))
>>                           == Forall([(D_82:
>>                              obj) => (((D_82
>>                              E (Sc(Sc(.M_1))
>>                              Set [(C_83:obj)
>>                                 => (thetachain1(.M_1,
>>                                 .thelaw_1,C_83):
>>                                 prop)]))
>>                              -> (C_81 E D_82)):
>>                              prop)]))
>>                           :prop)])
>>                        Fixform (((Misset_1
>>                        thetascm2 thelawchooses_1)
>>                        Iff2 (Sc(.M_1) Ui Ug([(C_90:
>>                           obj) => (Dediff([(hyp2_92:
>>                              that (C_90 E (Sc(Sc(.M_1))
>>                              Set [(C_93:obj)
>>                                 => (thetachain1(.M_1,
>>                                 .thelaw_1,C_93):
>>                                 prop)]))
>>                              ) => (Simp2((hyp2_92
>>                              Iff1 (C_90 Ui
>>                              (Sc(Sc(.M_1))
>>                              Separation [(C_97:
>>                                 obj) => (thetachain1(.M_1,
>>                                 .thelaw_1,C_97):
>>                                 prop)]))
>>                              )):that thetachain1(.M_1,
>>                              .thelaw_1,C_90))]
>>                           ,[(hyp1_98:that thetachain1(.M_1,
>>                              .thelaw_1,C_90))
>>                              => (((C_90 E (Sc(Sc(.M_1))
>>                              Set [(C_99:obj)
>>                                 => (thetachain1(.M_1,
>>                                 .thelaw_1,C_99):
>>                                 prop)]))
>>                              Fixform (((Simp1(Simp2(hyp1_98))
>>                              Iff2 (C_90 Ui
>>                              Scthm(Sc(.M_1))))
>>                              Conj hyp1_98)
>>                              Iff2 (C_90 Ui
>>                              (Sc(Sc(.M_1))
>>                              Separation [(C_110:
>>                                 obj) => (thetachain1(.M_1,
>>                                 .thelaw_1,C_110):
>>                                 prop)]))
>>                              )):that (C_90
>>                              E (Sc(Sc(.M_1))
>>                              Set [(C_111:obj)
>>                                 => (thetachain1(.M_1,
>>                                 .thelaw_1,C_111):
>>                                 prop)]))
>>                              )])
>>                           :that ((C_90 E (Sc(Sc(.M_1))
>>                           Set [(C_112:obj)
>>                              => (thetachain1(.M_1,
>>                              .thelaw_1,C_112):
>>                              prop)]))
>>                           == thetachain1(.M_1,
>>                           .thelaw_1,C_90))]))
>>                        ) Mp ((Sc(Sc(.M_1))
>>                        Set [(C_117:obj) =>
>>                           (thetachain1(.M_1,
>>                           .thelaw_1,C_117):
>>                           prop)])
>>                        Intax Sc(.M_1)))))))):
```

```
>>                        that (J_9 E K_15))]))
>>                     )))) Conj Simp2(thehyp_10))
>>                     Mp (J_9 Ui (H_7 Ui Simp2(Simp2(Simp2((ktheta_17
>>                     Iff1 (K_15 Ui Ug([(C_135:
>>                        obj) => (Dediff([(hyp2_137:
>>                           that (C_135 E (Sc(Sc(.M_1))
>>                           Set [(C_138:obj)
>>                              => (thetachain1(.M_1,
>>                              .thelaw_1,C_138):
>>                              prop)]))
>>                           ) => (Simp2((hyp2_137
>>                           Iff1 (C_135 Ui (Sc(Sc(.M_1))
>>                           Separation [(C_142:
>>                              obj) => (thetachain1(.M_1,
>>                              .thelaw_1,C_142):
>>                              prop)]))
>>                           )):that thetachain1(.M_1,
>>                           .thelaw_1,C_135))]
>>                        ,[(hyp1_143:that thetachain1(.M_1,
>>                           .thelaw_1,C_135))
>>                           => (((C_135 E (Sc(Sc(.M_1))
>>                           Set [(C_144:obj)
>>                              => (thetachain1(.M_1,
>>                              .thelaw_1,C_144):
>>                              prop)]))
>>                           Fixform (((Simp1(Simp2(hyp1_143))
>>                           Iff2 (C_135 Ui Scthm(Sc(.M_1))))
>>                           Conj hyp1_143) Iff2
>>                           (C_135 Ui (Sc(Sc(.M_1))
>>                           Separation [(C_155:
>>                              obj) => (thetachain1(.M_1,
>>                              .thelaw_1,C_155):
>>                              prop)]))
>>                           )):that (C_135 E
>>                           (Sc(Sc(.M_1)) Set
>>                           [(C_156:obj) => (thetachain1(.M_1,
>>                              .thelaw_1,C_156):
>>                              prop)]))
>>                              )])
>>                           :that ((C_135 E (Sc(Sc(.M_1))
>>                           Set [(C_157:obj) =>
>>                              (thetachain1(.M_1,
>>                              .thelaw_1,C_157):
>>                              prop)]))
>>                           == thetachain1(.M_1,
>>                           .thelaw_1,C_135)))]))
>>                        ))))))):that ((H_7 Intersection
>>                        J_9) E K_15))])
>>                     :that ((K_15 E (Sc(Sc(.M_1))
>>                     Set [(C_158:obj) => (thetachain1(.M_1,
>>                        .thelaw_1,C_158):prop)]))
>>                     -> ((H_7 Intersection J_9)
>>                     E K_15)))])
>>                  Iff2 ((H_7 Intersection J_9)
>>                  Ui (Forall([(C_162:obj) => (((C_162
>>                     E (Misset_1 Mbold2 thelawchooses_1))
>>                     == Forall([(D_163:obj) =>
>>                        (((D_163 E (Sc(Sc(.M_1))
>>                        Set [(C_164:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_164):prop)]))
>>                        -> (C_162 E D_163)):prop)]))
>>                     :prop)])
>>                  Fixform (((Misset_1 thetascm2
>>                  thelawchooses_1) Iff2 (Sc(.M_1)
>>                  Ui Ug([(C_171:obj) => (Dediff([(hyp2_173:
>>                        that (C_171 E (Sc(Sc(.M_1))
>>                        Set [(C_174:obj) => (thetachain1(.M_1,
>>                           .thelaw_1,C_174):prop)]))
>>                        ) => (Simp2((hyp2_173 Iff1
>>                        (C_171 Ui (Sc(Sc(.M_1))
>>                        Separation [(C_178:obj)
>>                           => (thetachain1(.M_1,
>>                           .thelaw_1,C_178):prop)]))
>>                        )):that thetachain1(.M_1,
>>                        .thelaw_1,C_171))]
>>                     ,[(hyp1_179:that thetachain1(.M_1,
>>                      .thelaw_1,C_171)) => (((C_171
>>                      E (Sc(Sc(.M_1)) Set [(C_180:
>>                         obj) => (thetachain1(.M_1,
>>                         .thelaw_1,C_180):prop)]))
>>                      Fixform (((Simp1(Simp2(hyp1_179))
>>                      Iff2 (C_171 Ui Scthm(Sc(.M_1))))
>>                      Conj hyp1_179) Iff2 (C_171
>>                      Ui (Sc(Sc(.M_1)) Separation
>>                      [(C_191:obj) => (thetachain1(.M_1,
>>                         .thelaw_1,C_191):prop)]))
>>                      )):that (C_171 E (Sc(Sc(.M_1))
>>                      Set [(C_192:obj) => (thetachain1(.M_1,
>>                         .thelaw_1,C_192):prop)]))
>>                      )])
>>                   :that ((C_171 E (Sc(Sc(.M_1))
>>                   Set [(C_193:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_193):prop)]))
>>                   == thetachain1(.M_1,.thelaw_1,
>>                   C_171)))]))
>>                ) Mp ((Sc(Sc(.M_1)) Set [(C_198:
>>                   obj) => (thetachain1(.M_1,
>>                   .thelaw_1,C_198):prop)])
>>                Intax Sc(.M_1)))))):that ((H_7
>>                Intersection J_9) E (Misset_1
>>                Mbold2 thelawchooses_1)))])
>>             :that (((H_7 <<= (Misset_1 Mbold2
>>             thelawchooses_1)) & (J_9 E H_7))
>>             -> ((H_7 Intersection J_9) E (Misset_1
>>             Mbold2 thelawchooses_1))))])
>>          :that Forall([(J_199:obj) => ((((H_7
>>             <<= (Misset_1 Mbold2 thelawchooses_1))
>>             & (J_199 E H_7)) -> ((H_7 Intersection
>>             J_199) E (Misset_1 Mbold2 thelawchooses_1))):
>>             prop)]))
>>          ])
>>       :that Forall([(H_200:obj) => (Forall([(J_201:
>>             obj) => ((((H_200 <<= (Misset_1
>>             Mbold2 thelawchooses_1)) & (J_201
>>             E H_200)) -> ((H_200 Intersection
>>             J_201) E (Misset_1 Mbold2 thelawchooses_1))):
>>             prop)])
>>          :prop)]))
>>       ]
>>    {move 0}
```

```
open

   define Line14: Lineb14 Misset thelawchooses


>>    Line14: [(---:that Forall([(H_1:obj) =>
>>            (Forall([(J_2:obj) => ((((H_1 <<=
>>               (Misset Mbold2 thelawchooses))
>>               & (J_2 E H_1)) -> ((H_1 Intersection
>>               J_2) E (Misset Mbold2 thelawchooses))):
>>               prop)])
>>            :prop)]))
>>         ]
>>      {move 1}
```

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

```
Lestrade execution:


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

>>    Mboldtheta1: [(---:that thetachain(Mbold))]
>>      {move 1}
```

```
   close

define Mboldtheta2 Misset thelawchooses: \
   Mboldtheta1

>> Mboldtheta2: [(.M_1:obj),(Misset_1:that Isset(.M_1)),
>>      (.thelaw_1:[(S_2:obj) => (---:obj)]),
>>      (thelawchooses_1:[(.S_3:obj),(subsetev_3:
>>         that (.S_3 <<= .M_1)),(inev_3:that
>>         Exists([(x_4:obj) => ((x_4 E .S_3):
>>            prop)]))
>>         => (---:that (.thelaw_1(.S_3) E .S_3))])
>>      => ((thetachain1(.M_1,.thelaw_1,(Misset_1
>>      Mbold2 thelawchooses_1)) Fixform ((Ug([(F_9:
>>         obj) => (Ded([(ftheta_11:that (F_9
>>            E (Sc(Sc(.M_1)) Set [(C_12:obj)
>>               => (thetachain1(.M_1,.thelaw_1,
>>               C_12):prop)]))
>>            ) => (Simp1((ftheta_11 Iff1 (F_9
>>            Ui Ug([(C_21:obj) => (Dediff([(hyp2_23:
>>                  that (C_21 E (Sc(Sc(.M_1))
>>                  Set [(C_24:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_24):prop)]))
>>                  ) => (Simp2((hyp2_23 Iff1
>>                  (C_21 Ui (Sc(Sc(.M_1)) Separation
>>                  [(C_28:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_28):prop)]))
>>                  )):that thetachain1(.M_1,.thelaw_1,
>>                  C_21))]
>>               ,[(hyp1_29:that thetachain1(.M_1,
>>                  .thelaw_1,C_21)) => (((C_21
>>                  E (Sc(Sc(.M_1)) Set [(C_30:
>>                     obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_30):prop)]))
>>                  Fixform (((Simp1(Simp2(hyp1_29))
>>                  Iff2 (C_21 Ui Scthm(Sc(.M_1))))
>>                    Conj hyp1_29) Iff2 (C_21 Ui
>>                    (Sc(Sc(.M_1)) Separation [(C_41:
>>                       obj) => (thetachain1(.M_1,
>>                       .thelaw_1,C_41):prop)]))
>>                    )):that (C_21 E (Sc(Sc(.M_1))
>>                    Set [(C_42:obj) => (thetachain1(.M_1,
>>                       .thelaw_1,C_42):prop)]))
>>                    )])
>>                 :that ((C_21 E (Sc(Sc(.M_1))
>>                 Set [(C_43:obj) => (thetachain1(.M_1,
>>                    .thelaw_1,C_43):prop)]))
>>                 == thetachain1(.M_1,.thelaw_1,
>>                 C_21)))]))
>>              )):that (.M_1 E F_9))])
>>           :that ((F_9 E (Sc(Sc(.M_1)) Set [(C_44:
>>              obj) => (thetachain1(.M_1,.thelaw_1,
>>              C_44):prop)]))
>>           -> (.M_1 E F_9)))])
>>        Iff2 (.M_1 Ui (Forall([(C_48:obj) => (((C_48
>>           E (Misset_1 Mbold2 thelawchooses_1))
>>           == Forall([(D_49:obj) => (((D_49 E
>>              (Sc(Sc(.M_1)) Set [(C_50:obj) =>
>>                 (thetachain1(.M_1,.thelaw_1,C_50):
>>                 prop)]))
>>              -> (C_48 E D_49)):prop)]))
>>           :prop)])
>>        Fixform (((Misset_1 thetascm2 thelawchooses_1)
>>        Iff2 (Sc(.M_1) Ui Ug([(C_57:obj) => (Dediff([(hyp2_59:
>>              that (C_57 E (Sc(Sc(.M_1)) Set [(C_60:
>>                 obj) => (thetachain1(.M_1,.thelaw_1,
>>                 C_60):prop)]))
>>              ) => (Simp2((hyp2_59 Iff1 (C_57
>>              Ui (Sc(Sc(.M_1)) Separation [(C_64:
>>                 obj) => (thetachain1(.M_1,.thelaw_1,
>>                 C_64):prop)]))
>>              )):that thetachain1(.M_1,.thelaw_1,
>>              C_57))]
>>           ,[(hyp1_65:that thetachain1(.M_1,.thelaw_1,
>>              C_57)) => (((C_57 E (Sc(Sc(.M_1))
>>              Set [(C_66:obj) => (thetachain1(.M_1,
>>                 .thelaw_1,C_66):prop)]))
>>              Fixform (((Simp1(Simp2(hyp1_65))
>>              Iff2 (C_57 Ui Scthm(Sc(.M_1))))
>>              Conj hyp1_65) Iff2 (C_57 Ui (Sc(Sc(.M_1))
>>              Separation [(C_77:obj) => (thetachain1(.M_1,
>>                 .thelaw_1,C_77):prop)]))
>>              )):that (C_57 E (Sc(Sc(.M_1)) Set
>>              [(C_78:obj) => (thetachain1(.M_1,
>>                 .thelaw_1,C_78):prop)]))
>>              )])
>>           :that ((C_57 E (Sc(Sc(.M_1)) Set [(C_79:
>>              obj) => (thetachain1(.M_1,.thelaw_1,
>>              C_79):prop)]))
>>           == thetachain1(.M_1,.thelaw_1,C_57))]))
>>        ) Mp ((Sc(Sc(.M_1)) Set [(C_84:obj) =>
>>           (thetachain1(.M_1,.thelaw_1,C_84):prop)])
>>        Intax Sc(.M_1)))))) Conj ((((Misset_1
>>        Mbold2 thelawchooses_1) <<= Sc(.M_1))
>>        Fixform (Ug([(A_90:obj) => (Ded([(ainev_91:
>>              that (A_90 E (Misset_1 Mbold2 thelawchooses_1)))
>>              => ((((Misset_1 thetascm2 thelawchooses_1)
>>              Iff2 (Sc(.M_1) Ui Ug([(C_98:obj)
>>                 => (Dediff([(hyp2_100:that (C_98
>>                    E (Sc(Sc(.M_1)) Set [(C_101:
>>                       obj) => (thetachain1(.M_1,
>>                       .thelaw_1,C_101):prop)]))
>>                    ) => (Simp2((hyp2_100 Iff1
>>                    (C_98 Ui (Sc(Sc(.M_1)) Separation
>>                    [(C_105:obj) => (thetachain1(.M_1,
>>                       .thelaw_1,C_105):prop)]))
>>                    )):that thetachain1(.M_1,.thelaw_1,
>>                    C_98))]
>>                 ,[(hyp1_106:that thetachain1(.M_1,
>>                    .thelaw_1,C_98)) => (((C_98
>>                    E (Sc(Sc(.M_1)) Set [(C_107:
>>                       obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_107):prop)]))
>>                   Fixform (((Simp1(Simp2(hyp1_106))
>>                   Iff2 (C_98 Ui Scthm(Sc(.M_1))))
>>                   Conj hyp1_106) Iff2 (C_98
>>                   Ui (Sc(Sc(.M_1)) Separation
>>                   [(C_118:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_118):prop)]))
>>                   )):that (C_98 E (Sc(Sc(.M_1))
>>                   Set [(C_119:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_119):prop)]))
>>                   )])
>>                :that ((C_98 E (Sc(Sc(.M_1))
>>                Set [(C_120:obj) => (thetachain1(.M_1,
>>                   .thelaw_1,C_120):prop)]))
>>                == thetachain1(.M_1,.thelaw_1,
>>                C_98)))]))
>>             ) Mp (Sc(.M_1) Ui (ainev_91 Iff1
>>             (A_90 Ui (Forall([(C_128:obj) =>
>>                (((C_128 E (Misset_1 Mbold2 thelawchooses_1))
>>                == Forall([(D_129:obj) => (((D_129
>>                   E (Sc(Sc(.M_1)) Set [(C_130:
>>                      obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_130):prop)]))
>>                   -> (C_128 E D_129)):prop)]))
>>                :prop)])
>>             Fixform (((Misset_1 thetascm2 thelawchooses_1)
>>             Iff2 (Sc(.M_1) Ui Ug([(C_137:obj)
>>                => (Dediff([(hyp2_139:that (C_137
>>                   E (Sc(Sc(.M_1)) Set [(C_140:
>>                      obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_140):prop)]))
>>                   ) => (Simp2((hyp2_139 Iff1
>>                   (C_137 Ui (Sc(Sc(.M_1)) Separation
>>                   [(C_144:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_144):prop)]))
>>                   )):that thetachain1(.M_1,.thelaw_1,
>>                   C_137))]
>>                ,[(hyp1_145:that thetachain1(.M_1,
>>                   .thelaw_1,C_137)) => (((C_137
>>                   E (Sc(Sc(.M_1)) Set [(C_146:
>>                      obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_146):prop)]))
>>                   Fixform (((Simp1(Simp2(hyp1_145))
>>                   Iff2 (C_137 Ui Scthm(Sc(.M_1))))
>>                   Conj hyp1_145) Iff2 (C_137
>>                   Ui (Sc(Sc(.M_1)) Separation
>>                   [(C_157:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_157):prop)]))
>>                   )):that (C_137 E (Sc(Sc(.M_1))
>>                   Set [(C_158:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_158):prop)]))
>>                   )])
>>                :that ((C_137 E (Sc(Sc(.M_1))
>>                Set [(C_159:obj) => (thetachain1(.M_1,
>>                   .thelaw_1,C_159):prop)]))
>>                == thetachain1(.M_1,.thelaw_1,
>>                C_137)))]))
>>             ) Mp ((Sc(Sc(.M_1)) Set [(C_164:
>>                obj) => (thetachain1(.M_1,.thelaw_1,
>>                C_164):prop)])
>>             Intax Sc(.M_1)))))))):that (A_90
>>             E Sc(.M_1)))])
>>          :that ((A_90 E (Misset_1 Mbold2 thelawchooses_1))
>>          -> (A_90 E Sc(.M_1))))])
>>       Conj (Inhabited((Ug([(F_169:obj) => (Ded([(ftheta_171:
>>             that (F_169 E (Sc(Sc(.M_1)) Set
>>             [(C_172:obj) => (thetachain1(.M_1,
>>                .thelaw_1,C_172):prop)]))
>>             ) => (Simp1((ftheta_171 Iff1 (F_169
>>             Ui Ug([(C_181:obj) => (Dediff([(hyp2_183:
>>                   that (C_181 E (Sc(Sc(.M_1))
>>                   Set [(C_184:obj) => (thetachain1(.M_1,
>>                      .thelaw_1,C_184):prop)]))
>>                   ) => (Simp2((hyp2_183 Iff1
>>                   (C_181 Ui (Sc(Sc(.M_1)) Separation
>>                   [(C_188:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_188):prop)]))
>>                  )):that thetachain1(.M_1,.thelaw_1,
>>                  C_181))]
>>               ,[(hyp1_189:that thetachain1(.M_1,
>>                  .thelaw_1,C_181)) => (((C_181
>>                  E (Sc(Sc(.M_1)) Set [(C_190:
>>                     obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_190):prop)]))
>>                  Fixform (((Simp1(Simp2(hyp1_189))
>>                  Iff2 (C_181 Ui Scthm(Sc(.M_1))))
>>                  Conj hyp1_189) Iff2 (C_181
>>                  Ui (Sc(Sc(.M_1)) Separation
>>                  [(C_201:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_201):prop)]))
>>                  )):that (C_181 E (Sc(Sc(.M_1))
>>                  Set [(C_202:obj) => (thetachain1(.M_1,
>>                     .thelaw_1,C_202):prop)]))
>>                  )])
>>               :that ((C_181 E (Sc(Sc(.M_1))
>>               Set [(C_203:obj) => (thetachain1(.M_1,
>>                  .thelaw_1,C_203):prop)]))
>>               == thetachain1(.M_1,.thelaw_1,
>>               C_181)))]))
>>            )):that (.M_1 E F_169))])
>>         :that ((F_169 E (Sc(Sc(.M_1)) Set [(C_204:
>>            obj) => (thetachain1(.M_1,.thelaw_1,
>>            C_204):prop)]))
>>         -> (.M_1 E F_169)))])
>>      Iff2 (.M_1 Ui (Forall([(C_208:obj) =>
>>         (((C_208 E (Misset_1 Mbold2 thelawchooses_1))
>>         == Forall([(D_209:obj) => (((D_209
>>            E (Sc(Sc(.M_1)) Set [(C_210:obj)
>>               => (thetachain1(.M_1,.thelaw_1,
>>               C_210):prop)]))
>>            -> (C_208 E D_209)):prop)]))
>>         :prop)])
>>      Fixform (((Misset_1 thetascm2 thelawchooses_1)
>>      Iff2 (Sc(.M_1) Ui Ug([(C_217:obj) => (Dediff([(hyp2_219:
```

```
>>            that (C_217 E (Sc(Sc(.M_1)) Set
>>            [(C_220:obj) => (thetachain1(.M_1,
>>               .thelaw_1,C_220):prop)]))
>>            ) => (Simp2((hyp2_219 Iff1 (C_217
>>            Ui (Sc(Sc(.M_1)) Separation [(C_224:
>>               obj) => (thetachain1(.M_1,.thelaw_1,
>>               C_224):prop)]))
>>            )):that thetachain1(.M_1,.thelaw_1,
>>            C_217))]
>>         ,[(hyp1_225:that thetachain1(.M_1,.thelaw_1,
>>            C_217)) => (((C_217 E (Sc(Sc(.M_1))
>>            Set [(C_226:obj) => (thetachain1(.M_1,
>>               .thelaw_1,C_226):prop)]))
>>            Fixform (((Simp1(Simp2(hyp1_225))
>>            Iff2 (C_217 Ui Scthm(Sc(.M_1))))
>>            Conj hyp1_225) Iff2 (C_217 Ui (Sc(Sc(.M_1))
>>            Separation [(C_237:obj) => (thetachain1(.M_1,
>>               .thelaw_1,C_237):prop)]))
>>            )):that (C_217 E (Sc(Sc(.M_1)) Set
>>            [(C_238:obj) => (thetachain1(.M_1,
>>               .thelaw_1,C_238):prop)]))
>>            )])
>>         :that ((C_217 E (Sc(Sc(.M_1)) Set [(C_239:
>>            obj) => (thetachain1(.M_1,.thelaw_1,
>>            C_239):prop)]))
>>         == thetachain1(.M_1,.thelaw_1,C_217))]))
>>      ) Mp ((Sc(Sc(.M_1)) Set [(C_244:obj) =>
>>         (thetachain1(.M_1,.thelaw_1,C_244):
>>         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)))]
>>   {move 0}
```

```
open

   define Mboldtheta: Mboldtheta2 Misset \
      thelawchooses

>>    Mboldtheta: [(---:that thetachain1(M,thelaw,
>>         (Misset Mbold2 thelawchooses)))]
>>      {move 1}
```

`Mboldtheta` asserts that **M** is a Θ-chain.