tail end of section 5, debugging

April 10, 2020

```
begin Lestrade execution

                  linea90 : that (((Misset
                   Mbold2 thelawchooses Set
                   [(x1_5 : obj) =>
                      ({def} S <<= x1_5 : prop)]) Intersection
                   M) E Misset Mbold2 thelawchooses) & (Usc
                   (chosenof (S)) <<=
                   Rcal1 (S)) & chosenof
                   (S) E chosenof (S) ; chosenof
                   (S)


                  {move 5}

                  >>> define line90 : Fixform \
                      (Rcal1 S = Rcal chosenof \
                      S, Line41 (Iff2 (Mpsubs \
                      line85 Ssubm zins Ssubm, Uscsubs \
                      (chosenof S, M)), Pairinhabited \
                      (chosenof S, chosenof \
                      S), linea90))

objectof a non object line 2749: {function error}

(paused, type something to continue) >
objectof a non object line 2749: {function error}

(paused, type something to continue) >
objectof a non object line 2749: {function error}

(paused, type something to continue) >
general failure of functionsort line 3030
```

```
(paused, type something to continue) >
general failure of functionsort line 3030

(paused, type something to continue) >
Comparison failed of obj and {function error}

(paused, type something to continue) >
Object type error in Line41(Ssubm line85 zins Mpsubs Ssubm Iff2 chosenof(S) Uscsubs M, c

(paused, type something to continue) >
Typefix failure with application term

general failure of functionsort line 3030

(paused, type something to continue) >
general failure of functionsort line 3030

(paused, type something to continue) >
Comparison failed of that Rcal1(S) = Rcal(chosenof(S)) and {function error}

(paused, type something to continue) >
Object type error in (Rcal1(S) = Rcal(chosenof(S))) Fixform {function error}

(paused, type something to continue) >
Typefix failure with application term

implicitarglist failure line 1905

(paused, type something to continue) >
Parse or typefix error in(Rcal1(S) = Rcal(chosenof(S))) Fixform Line41(chosenof(S) Pairi

(paused, type something to continue) >

                  >>> define line91 : Subs1 \
                      line90, Mpsubs thehyp, Linea13 \
                      Ssubm, Ei1 z zins

Subs1 line90, Mpsubs thehyp, Linea13 Ssubm, Ei1 z zins is not well-formed

(paused, type something to continue) >

                  >>> define line92 case2 \
                      : Fixform (chosenof S <~ \
                      xx, (Mpsubs line85 Ssubm \
                      zins Ssubm) Conj (Mpsubs \
                      thehyp Ssubm) Conj (Negeqsymm \
```

```
                    case2) Conj line91)

[case2 => Fixform (chosenof S <~ xx, (Mpsubs line85 Ssubm zins Ssubm) Conj (Mpsubs thehy

(paused, type something to continue) >

                >>> define line93 case2 \
                    : Add2 (xx = chosenof \
                    S, line92 case2)

[case2 => Add2 (xx = chosenof S, line92 case2)] is not well-formed

(paused, type something to continue) >

                >>> close


              {move 5}

              >>> define line94 thehyp : Cases \
                  line86 thehyp, line87, line93

[thehyp => Cases line86 thehyp, line87, line93] is not well-formed

(paused, type something to continue) >

              >>> close


           {move 4}

           >>> define line95 xx : Ded line94

[xx => Ded line94] is not well-formed

(paused, type something to continue) >

           >>> close


        {move 3}

        >>> define line96 Ssubm zins : Ug \
            line95

[Ssubm zins => Ug line95] is not well-formed
```

```
(paused, type something to continue) >

         >>> define line97 Ssubm zins : Ei1 \
             chosenof S, Conj (line85 Ssubm \
             zins, line96 Ssubm zins)

[Ssubm zins => Ei1 chosenof S, Conj (line85 Ssubm zins, line96 Ssubm zins)] is not well-

(paused, type something to continue) >

         >>> open


            {move 4}

            >>> declare x66 obj


            x66 : obj


            {move 4}

            >>> declare thehyp that (S <<= \
                M) & Exists [x66 => x66 E S]


            thehyp : that (S <<= M) & Exists
             ([(x66_3 : obj) =>
                ({def} x66_3 E S : prop)])


            {move 4}

            >>> open


               {move 5}

               >>> declare y66 obj


               y66 : obj
```

```
               {move 5}

               >>> declare yins66 that y66 \
                   E S


               yins66 : that y66 E S


               {move 5}

               >>> define line98 yins66 : line97 \
                   Simp1 thehyp yins66

[yins66 => line97 Simp1 thehyp yins66] is not well-formed

(paused, type something to continue) >

               >>> close


            {move 4}

            >>> define line99 thehyp : Eg \
                Simp2 thehyp line98

[thehyp => Eg Simp2 thehyp line98] is not well-formed

(paused, type something to continue) >

            >>> close


         {move 3}

         >>> define line10 S : Ded line99

[S => Ded line99] is not well-formed

(paused, type something to continue) >

         >>> close


      {move 2}
```

```
      >>> define line11 : Ug line10

Ug line10 is not well-formed

(paused, type something to continue) >

      >>> close


   {move 1}

   >>> comment the following line will not \
       run until we work on definition expansion \
       control in the text above


   {move 1}

   >>> define line12 Misset thelawchooses \
       : line11

[Misset thelawchooses => line11] is not well-formed

(paused, type something to continue) >
end Lestrade execution
```

We prove that a nonempty subset S of M has a minimal element in the order. The minimal element is the distinguished element s of $\mathcal{R}_1(S)$. One shows that $\mathcal{R}_1(S) = \mathcal{R}(s)$, from which it follows readily that s is an element of S and minimal in the order we defined.

This completes the proof that if we have a method of choosing a distinguished element from each subset of M, we can well-order M.

It remains to show that the Axiom of Choice in its usual form allows us to choose distinguished elements as required.