Implementation of Zermelo's work of 1908 in Lestrade: Part IV, central impredicative argument for total ordering of $\mathbf{M}$

M. Randall Holmes

January 5, 2022

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.

This particular part is monstrously large and slow and needs some fine tuning.

In this section, we prove that $\mathbf{M}$ is totally ordered by inclusion. This involves showing that the collection of elements of $\mathbf{M}$ which either include or are included in each other element of $\mathbf{M}$ is itself a Θ-chain and so actually equal to $\mathbf{M}$. The horrible thing about this is that the proof of the third component of this result contains a proof that a further refinement of this set definition also yields a Θ-chain, with its own four parts.

```
begin Lestrade execution
```

```
      >>> comment load whatismath3

      {move 2}

      >>> clearcurrent
{move 2}

      >>> declare C obj

      C : obj

      {move 2}

      >>> declare D obj

      D : obj

      {move 2}

      >>> define cuts1 C : (C E Mbold) & Forall \
          [D => (D E Mbold) -> (D <<= C) V (C <<= \
             D)]

      cuts1 : [(C_1 : obj) => (--- : prop)]

      {move 1}

      >>> save

      {move 2}

      >>> close

   {move 1}

   >>> declare C666 obj
```

```
C666 : obj

{move 1}

>>> define cuts2 Misset, thelawchooses, C666 \
    : cuts1 C666

cuts2 : [(.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)]), (C666_1
    : obj) =>
    ({def} (C666_1 E Misset_1 Mbold2
    thelawchooses_1) & Forall ([(D_3
       : obj) =>
       ({def} (D_3 E Misset_1 Mbold2
       thelawchooses_1) -> (D_3 <<= C666_1) V C666_1
       <<= D_3 : prop)]) : prop)]

cuts2 : [(.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)]), (C666_1
    : obj) => (--- : prop)]

{move 0}

>>> open

   {move 2}
```

```
   >>> define cuts C : cuts2 Misset, thelawchooses, C

   cuts : [(C_1 : obj) => (--- : prop)]

   {move 1}

   >>> define Cuts1 : Set (Mbold, cuts)

   Cuts1 : obj

   {move 1}

   >>> close

{move 1}

>>> define Cuts3 Misset thelawchooses \
    : Cuts1

Cuts3 : [(.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} Misset_1 Mbold2 thelawchooses_1
    Set [(C_2 : obj) =>
       ({def} cuts2 (Misset_1, thelawchooses_1, C_2) : prop)] : obj)]

Cuts3 : [(.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)]) =>
      (--- : obj)]

   {move 0}

   >>> open

      {move 2}

      >>> define Cuts : Cuts3 Misset, thelawchooses

      Cuts : obj

      {move 1}
end Lestrade execution
```

This defines the predicate "is an element of **M** which either includes or is included in each element of `M`" and the correlated set. These things are packaged so as not to expand. The aim is to show that `Cuts` is a Θ-chain, from which we will be able to show the desired linear ordering result.

```
begin Lestrade execution

      >>> define line1 : Simp1 Mboldtheta

      line1 : that M E Misset Mbold2 thelawchooses

      {move 1}

      >>> 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}

   >>> define line2 finmbold : Iff1 \
       (Mp finmbold, Ui F Simp1 Simp1 \
       Simp2 Mboldtheta, Ui F Scthm \
       M)

   line2 : [(finmbold_1 : that
       F E Mbold) => (--- : that
       F <<= M)]

   {move 3}

   >>> define line3 finmbold : Add1 \
       (M <<= F, line2 finmbold)

   line3 : [(finmbold_1 : that
       F E Mbold) => (--- : that
       (F <<= M) V M <<= F)]

   {move 3}

   >>> close

{move 3}

>>> define line4 F : Ded line3
```

```
   line4 : [(F_1 : obj) => (---
       : that (F_1 E Mbold) -> (F_1
       <<= M) V M <<= F_1)]

   {move 2}

   >>> close

{move 2}

>>> define line5 : Ug line4

line5 : that Forall ([(x'_2 : obj) =>
    ({def} (x'_2 E Mbold) -> (x'_2
    <<= M) V M <<= x'_2 : prop)])

{move 1}

>>> define line6 : Fixform (cuts M, Conj \
    (line1, line5))

line6 : that cuts (M)

{move 1}

>>> define line7 : Conj (Simp1 Mboldtheta, line6)

line7 : that (M E Misset Mbold2 thelawchooses) & cuts
 (M)

{move 1}

>>> define line8 : Ui M, Separation \
    (Mbold, cuts)

line8 : that (M E Mbold Set cuts) ==
 (M E Mbold) & cuts (M)
```

```
      {move 1}

      >>> define Line9 : Fixform (M E Cuts, Iff2 \
          (line7, line8))

      Line9 : that M E Cuts

      {move 1}
end Lestrade execution
```

This is the first component of the proof that `Cuts` is a Θ-chain.

```
begin Lestrade execution

      >>> define line10 : Fixform (Cuts \
          <<= (Mbold), Sepsub (Mbold, cuts, Inhabited \
          (Simp1 (Mboldtheta))))

      line10 : that Cuts <<= Mbold

      {move 1}

      >>> define line11 : Fixform ((Mbold) <<= \
          Sc M, Sepsub2 (Sc2 M, Refleq (Mbold)))

      line11 : that Mbold <<= Sc (M)

      {move 1}

      >>> define Line12 : Transsub (line10, line11)

      Line12 : that Cuts <<= Sc (M)

      {move 1}
end Lestrade execution
```

This is the second component of the proof that `Cuts` is a Θ-chain.

```
begin Lestrade execution

      >>> open

         {move 3}

         >>> declare B obj

         B : obj

         {move 3}

         >>> open

            {move 4}

            >>> declare bhyp that B E Cuts

            bhyp : that B E Cuts

            {move 4}

            >>> define line13 bhyp : Iff1 \
                (bhyp, Ui B, Separation (Mbold, cuts))

            line13 : [(bhyp_1 : that B E Cuts) =>
                (--- : that (B E Mbold) & cuts
                (B))]

            {move 3}

            >>> define line14 bhyp : Simp1 \
                line13 bhyp

            line14 : [(bhyp_1 : that B E Cuts) =>
                (--- : that B E Mbold)]
```

```
{move 3}

>>> define linea14 bhyp : Setsinchains \
    Mboldtheta, line14 bhyp

linea14 : [(bhyp_1 : that B E Cuts) =>
    (--- : that Isset (B))]

{move 3}

>>> define lineb14 bhyp : Iff1 \
    (Mp (line14 bhyp, Ui (B, Simp1 \
    Simp1 Simp2 Mboldtheta)), Ui \
    B, Scthm M)

lineb14 : [(bhyp_1 : that B E Cuts) =>
    (--- : that B <<= M)]

{move 3}

>>> define line15 bhyp : Simp2 \
    Simp2 line13 bhyp

line15 : [(bhyp_1 : that B E Cuts) =>
    (--- : that Forall ([(D_2
       : obj) =>
       ({def} (D_2 E Misset
       Mbold2 thelawchooses) ->
       (D_2 <<= B) V B <<= D_2
       : prop)]))]

{move 3}

>>> open

   {move 5}

   >>> declare F obj
```

```
F : obj

{move 5}

>>> declare fhyp that F E (Mbold)

fhyp : that F E Mbold

{move 5}

>>> define line16 fhyp : Fixform \
    ((prime F) <<= F, Sepsub2 \
    (Setsinchains Mboldtheta, fhyp, Refleq \
    (prime F)))

line16 : [(.F_1 : obj), (fhyp_1
    : that .F_1 E Mbold) =>
    (--- : that prime (.F_1) <<=
    .F_1)]

{move 4}

>>> declare Y obj

Y : obj

{move 5}

>>> define cutsa2 Y : (Y <<= \
   prime B) V B <<= Y

cutsa2 : [(Y_1 : obj) =>
    (--- : prop)]

{move 4}

>>> save
```

```
      {move 5}

      >>> close

   {move 4}

   >>> declare Y10 obj

   Y10 : obj

   {move 4}

   >>> define cutsb2 Y10 : cutsa2 \
       Y10

   cutsb2 : [(Y10_1 : obj) =>
       (--- : prop)]

   {move 3}

   >>> save

   {move 4}

   >>> close

{move 3}

>>> declare Y11 obj

Y11 : obj

{move 3}

>>> define cutsc2 B Y11 : cutsb2 \
    Y11
```

```
   cutsc2 : [(B_1 : obj), (Y11_1
       : obj) => (--- : prop)]

   {move 2}

   >>> save

   {move 3}

   >>> close

{move 2}

>>> declare Ba1 obj

Ba1 : obj

{move 2}

>>> declare Y12 obj

Y12 : obj

{move 2}

>>> define cutsd2 Ba1 Y12 : cutsc2 \
    Ba1 Y12

cutsd2 : [(Ba1_1 : obj), (Y12_1
    : obj) => (--- : prop)]

{move 1}

>>> save

{move 2}

>>> close
```

```
{move 1}

>>> declare Ba2 obj

Ba2 : obj

{move 1}

>>> declare Y13 obj

Y13 : obj

{move 1}

>>> define cutse2 Misset, thelawchooses, Ba2 \
    Y13 : cutsd2 Ba2 Y13

cutse2 : [(.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)]), (Ba2_1
    : obj), (Y13_1 : obj) =>
    ({def} (Y13_1 <<= prime2 (.thelaw_1, Ba2_1)) V Ba2_1
    <<= Y13_1 : prop)]

cutse2 : [(.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)]), (Ba2_1
```

```
     : obj), (Y13_1 : obj) => (---
     : prop)]

{move 0}

>>> open

   {move 2}

   >>> define cutsf2 Ba1 Y12 : cutse2 \
       Misset, thelawchooses, Ba1 Y12

   cutsf2 : [(Ba1_1 : obj), (Y12_1
       : obj) => (--- : prop)]

   {move 1}

   >>> open

      {move 3}

      >>> define cutsg2 B Y11 : cutsf2 \
          B Y11

      cutsg2 : [(B_1 : obj), (Y11_1
          : obj) => (--- : prop)]

      {move 2}

      >>> open

         {move 4}

         >>> define cutsh2 Y10 : cutsg2 \
             B Y10

         cutsh2 : [(Y10_1 : obj) =>
             (--- : prop)]
```

```
            {move 3}

            >>> open

               {move 5}

               >>> define cutsi2 Y : cutsh2 \
                   Y

               cutsi2 : [(Y_1 : obj) =>
                   (--- : prop)]

               {move 4}

               >>> define Cuts2 : Set (Mbold, cutsi2)

               Cuts2 : obj

               {move 4}
end Lestrade execution
```

We are in the midst of the third component of the proof that `Cuts` is a Θ-chain. We have B which we assume is in `Cuts` and we want to show that `prime(B)` is in `Cuts`. We do this by showing that the set of all elements of `M` which are either included in `prime(B)` or include `B` is a Θ-chain. Thus we have four components of this proof to generate before we get to generating the third component of the proof for `Cuts`.

This is about the time that I defined the `goal` command which is used to generate helpful comments about what we are trying to prove in the rest of the files. I should probably backtrack and insert goal statements earlier!

```
begin Lestrade execution

               >>> goal that thetachain Cuts2

               that thetachain (Cuts2)
```

```
{move 5}

>>> comment test thetachain

{move 5}

>>> goal that M E Cuts2

that M E Cuts2

{move 5}

>>> define line17 : Ui M, Separation4 \
    Refleq Cuts2

line17 : that (M E Mbold
 Set cutsi2) == (M E Mbold) & cutsi2
 (M)

{move 4}

>>> define line18 : Conj (Simp1 \
    Mboldtheta, Add2 (M <<= \
    prime B, lineb14 bhyp))

line18 : that (M E Misset
 Mbold2 thelawchooses) & (M <<=
 prime (B)) V B <<= M

{move 4}

>>> define line19 : Fixform \
    (M E Cuts2, Iff2 line18 \
    line17)

line19 : that M E Cuts2
```

```
                {move 4}
end Lestrade execution
```

This is the first component of the proof that `Cuts2` is a Θ-chain.

```
begin Lestrade execution

                >>> goal that Cuts2 <<= Sc \
                    M

                that Cuts2 <<= Sc (M)

                {move 5}

                >>> declare D1 obj

                D1 : obj

                {move 5}

                >>> define line20 : Fixform \
                    (Cuts2 <<= Mbold, Sepsub2 \
                    (Separation3 Refleq Mbold, Refleq \
                    Cuts2))

                line20 : that Cuts2 <<= Mbold

                {move 4}

                >>> define line21 : Transsub \
                    line20 Simp1 Simp2 Mboldtheta

                line21 : that Cuts2 <<= Sc
                 (M)

                {move 4}
end Lestrade execution
```

This is the second component of the proof that `Cuts` is a Θ-chain.

```
begin Lestrade execution

               >>> declare F1 obj

               F1 : obj

               {move 5}

               >>> goal that Forall [D1 \
                      => (D1 E Cuts2) -> (prime \
                      D1) E Cuts2]

               that Forall ([(D1 : obj) =>
                   ({def} (D1 E Cuts2) ->
                   prime (D1) E Cuts2 : prop)])

               {move 5}

               >>> open

                  {move 6}

                  >>> declare D2 obj

                  D2 : obj

                  {move 6}

                  >>> open

                     {move 7}

                     >>> declare dhyp that \
                         D2 E Cuts2
```

```
dhyp : that D2 E Cuts2

{move 7}

>>> goal that (prime \
    D2) E Cuts2

that prime (D2) E Cuts2

{move 7}

>>> define line22 : Ui \
    prime D2, Separation4 \
    Refleq Cuts2

line22 : that (prime
 (D2) E Mbold Set cutsi2) ==
 (prime (D2) E Mbold) & cutsi2
 (prime (D2))

{move 6}

>>> goal that ((prime \
    D2) E Mbold) & ((prime \
    D2) <<= prime B) V (B <<= \
    prime D2)

that (prime (D2) E Mbold) & (prime
 (D2) <<= prime (B)) V B <<=
 prime (D2)

{move 7}

>>> define line23 dhyp \
    : Iff1 dhyp, Ui D2 \
    Separation4 Refleq Cuts2

line23 : [(dhyp_1
```

```
    : that D2 E Cuts2) =>
    (--- : that (D2
    E Mbold) & cutsi2
    (D2))]

{move 6}

>>> define line24 dhyp \
    : Simp1 line23 dhyp

line24 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that D2 E Mbold)]

{move 6}

>>> define line25 dhyp \
    : Simp2 line23 dhyp

line25 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that cutsi2
    (D2))]

{move 6}

>>> define line26 : Iff1 \
    bhyp, Ui B, Separation4 \
    Refleq Cuts

line26 : that (B E Misset
 Mbold2 thelawchooses) & cuts2
 (Misset, thelawchooses, B)

{move 6}

>>> define line27 dhyp \
    : Mp line24 dhyp, Ui \
```

```
    D2, Simp2 Simp2 line26

line27 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that (D2
    <<= B) V B <<= D2)]

{move 6}

>>> define line28 dhyp \
    : Mp line24 dhyp, Ui \
    D2, Simp1 Simp2 Simp2 \
    Mboldtheta

line28 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that prime2
    ([(S'_3 : obj) =>
       ({def} thelaw
       (S'_3) : obj)], D2) E Misset
    Mbold2 thelawchooses)]

{move 6}

>>> define line29 dhyp \
    : Mp line28 dhyp, Ui \
    prime D2, Simp2 Simp2 \
    line26

line29 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that (prime
    (D2) <<= B) V B <<=
    prime (D2))]

{move 6}

>>> goal that ((prime \
```

```
    D2) <<= prime B) V (B <<= \
    prime D2)

that (prime (D2) <<=
 prime (B)) V B <<=
 prime (D2)

{move 7}

>>> open

   {move 8}

   >>> declare U obj

   U : obj

   {move 8}

   >>> declare Casehyp1 \
       that B = 0

   Casehyp1 : that B = 0

   {move 8}

   >>> define linea29 \
       Casehyp1 : Subs1 \
       (Eqsymm Casehyp1, Add2 \
       (prime D2 <<= prime \
       B, (Zeroissubset \
       Separation3 Refleq \
       prime D2)))

   linea29 : [(Casehyp1_1
       : that B = 0) =>
       (--- : that (prime
       (D2) <<= prime
```

```
    (B)) V B <<=
    D2 Set [(x_4
       : obj) =>
       ({def} ~ (x_4
       E Usc (thelaw
       (D2))) : prop)])]

{move 7}

>>> declare Casehyp2 \
    that Exists [U => \
       U E B]

Casehyp2 : that Exists
 ([(U_2 : obj) =>
    ({def} U_2 E B : prop)])

{move 8}

>>> open

   {move 9}

   >>> declare casehyp1 \
       that D2 <<= prime \
       B

   casehyp1 : that
    D2 <<= prime (B)

   {move 9}

   >>> declare casehyp2 \
       that B <<= D2

   casehyp2 : that
    B <<= D2
```

```
{move 9}

>>> define line30 \
    casehyp1 : Transsub \
    (line16 (line24 \
    dhyp), casehyp1)

line30 : [(casehyp1_1
    : that D2 <<=
    prime (B)) =>
    (--- : that
    prime (D2) <<=
    prime (B))]

{move 8}

>>> define linea30 \
    casehyp1 : Add1 \
    (B <<= prime \
    D2, line30 casehyp1)

linea30 : [(casehyp1_1
    : that D2 <<=
    prime (B)) =>
    (--- : that
    (prime (D2) <<=
    prime (B)) V B <<=
    prime (D2))]

{move 8}

>>> define line31 \
    : Excmid ((thelaw \
    D2) = thelaw \
    B)

line31 : that
 (thelaw (D2) = thelaw
```

```
 (B)) V ~ (thelaw
 (D2) = thelaw
 (B))

{move 8}

>>> define line32 \
    : Separation4 \
    Refleq prime D2

line32 : that
 Forall ([(x_2
    : obj) =>
    ({def} (x_2
    E D2 Set [(x_5
       : obj) =>
       ({def} ~ (x_5
       E Usc (thelaw
       (D2))) : prop)]) ==
    (x_2 E D2) & ~ (x_2
    E Usc (thelaw
    (D2))) : prop)])

{move 8}

>>> open

   {move 10}

   >>> declare \
       casehypa1 that \
       (thelaw D2 \
       = thelaw B)

   casehypa1 : that
    thelaw (D2) = thelaw
    (B)
```

```
{move 10}

>>> declare \
    casehypa2 that \
    ~ (thelaw \
    D2 = thelaw \
    B)

casehypa2 : that
 ~ (thelaw
 (D2) = thelaw
 (B))

{move 10}

>>> open

   {move 11}

   >>> declare \
       G obj

   G : obj

   {move 11}

   >>> open

      {move
       12}

      >>> declare \
          onedir \
          that \
          G E prime \
          D2

      onedir
```

```
 : that
 G E prime
 (D2)

{move
 12}

>>> define \
    line33 \
    onedir \
    : Iff1 \
    onedir, Ui \
    G line32

line33
 : [(onedir_1
    : that
    G E prime
    (D2)) =>
    (---
    : that
    (G E D2) & ~ (G E Usc
    (thelaw
    (D2))))]

{move
 11}

>>> define \
    line34 \
    onedir \
    : Simp1 \
    line33 \
    onedir

line34
 : [(onedir_1
    : that
```

```
    G E prime
    (D2)) =>
    (---
    : that
    G E D2)]

{move
 11}

>>> define \
    line35 \
    onedir \
    : Simp2 \
    line33 \
    onedir

line35
 : [(onedir_1
    : that
    G E prime
    (D2)) =>
    (---
    : that
    ~ (G E Usc
    (thelaw
    (D2))))]

{move
 11}

>>> open

   {move
    13}

   >>> \
       declare \
       eqhyp \
```

```
    that \
    G = (thelaw \
    D2)

eqhyp
 : that
 G = thelaw
 (D2)

{move
 13}

>>> \
    define \
    line36 \
    eqhyp \
    : Subs1 \
    Eqsymm \
    eqhyp \
    line35 \
    onedir

line36
 : [(eqhyp_1
    : that
    G = thelaw
    (D2)) =>
    (---
    : that
    ~ (G E Usc
    (G)))]

{move
 12}

>>> \
    define \
    line37 \
```

```
            eqhyp \
            : Mp \
            (Inusc2 \
            G, line36 \
            eqhyp)

        line37
         : [(eqhyp_1
            : that
            G = thelaw
            (D2)) =>
            (---
            : that
            ??)]

        {move
         12}

        >>> \
            close

    {move
     12}

    >>> define \
        line38 \
        onedir \
        : Negintro \
        line37

    line38
     : [(onedir_1
        : that
        G E prime
        (D2)) =>
        (---
        : that
        ~ (G = thelaw
```

```
    (D2)))]

{move
 11}

>>> define \
    line39 \
    onedir \
    : Subs1 \
    casehypa1 \
    line38 \
    onedir

line39
 : [(onedir_1
    : that
    G E prime
    (D2)) =>
    (---
    : that
    ~ (G = thelaw
    (B)))]

{move
 11}

>>> define \
    linea39 \
    onedir \
    : Subs1 \
    casehypa1 \
    line35 \
    onedir

linea39
 : [(onedir_1
    : that
    G E prime
```

```
    (D2)) =>
    (---
    : that
    ~ (G E Usc
    (thelaw
    (B))))]

{move
 11}

>>> open

   {move
    13}

   >>> \
       declare \
       casehypb1 \
       that \
       prime \
       D2 \
       <<= \
       B

   casehypb1
    : that
    prime
    (D2) <<=
    B

   {move
    13}

   >>> \
       define \
       line40 \
       casehypb1 \
       : Mp \
```

```
    (onedir, Ui \
    G, Simp1 \
    casehypb1)

line40
 : [(casehypb1_1
    : that
    prime
    (D2) <<=
    B) =>
    (---
    : that
    G E B)]

{move
 12}

>>> \
    declare \
    casehypb2 \
    that \
    B <<= \
    prime \
    D2

casehypb2
 : that
 B <<=
 prime
 (D2)

{move
 13}

>>> \
    define \
    line41 \
    casehypb2 \
```

```
    : Ui \
    thelaw \
    B, Simp1 \
    casehypb2

line41
 : [(casehypb2_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    (thelaw
    (B) E B) ->
    thelaw
    (B) E prime
    (D2))]

{move
 12}

>>> \
    define \
    line42 \
    : thelawchooses \
    (lineb14 \
    bhyp, Casehyp2)

line42
 : that
 thelaw
 (B) E B

{move
 12}

>>> \
```

```
    define \
    line43 \
    casehypb2 \
    : Mp \
    (line42, line41 \
    casehypb2)

line43
 : [(casehypb2_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    thelaw
    (B) E prime
    (D2))]

{move
 12}

>>> \
    define \
    line44 \
    casehypb2 \
    : Iff1 \
    (line43 \
    casehypb2, Ui \
    thelaw \
    B, Separation4 \
    Refleq \
    prime \
    D2)

line44
 : [(casehypb2_1
    : that
```

```
    B <<=
    prime
    (D2)) =>
    (---
    : that
    (thelaw
    (B) E D2) & ~ (thelaw
    (B) E Usc
    (thelaw
    (D2))))]

{move
 12}

>>> \
    define \
    line45 \
    casehypb2 \
    : Subs1 \
    Eqsymm \
    casehypa1 \
    line44 \
    casehypb2

line45
 : [(casehypb2_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    (thelaw
    (D2) E D2) & ~ (thelaw
    (D2) E Usc
    (thelaw
    (D2))))]
```

```
{move
 12}

>>> \
    define \
    line46 \
    casehypb2 \
    : Simp2 \
    line45 \
    casehypb2

line46
 : [(casehypb2_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    ~ (thelaw
    (D2) E Usc
    (thelaw
    (D2))))]

{move
 12}

>>> \
    define \
    line47 \
    casehypb2 \
    : Giveup \
    (G E B, Mp \
    (Inusc2 \
    thelaw \
    D2, line46 \
    casehypb2))
```

```
   line47
    : [(casehypb2_1
       : that
       B <<=
       prime
       (D2)) =>
       (---
       : that
       G E B)]

   {move
    12}

   >>> \
       close

{move
 12}

>>> define \
    line48 \
    onedir \
    : Cases \
    (line29 \
    dhyp, line40, line47)

line48
 : [(onedir_1
    : that
    G E prime
    (D2)) =>
    (---
    : that
    G E B)]

{move
 11}
```

```
>>> define \
    linea48 \
    onedir \
    : Fixform \
    (G E prime \
    (B), Iff2 \
    (Conj \
    (line48 \
    onedir, linea39 \
    onedir), Ui \
    G, Separation4 \
    Refleq \
    prime \
    B))

linea48
 : [(onedir_1
    : that
    G E prime
    (D2)) =>
    (---
    : that
    G E prime
    (B))]

{move
 11}

>>> declare \
    otherdir \
    that \
    G E B

otherdir
 : that
 G E B

{move
```

```
 12}

>>> define \
    line49 \
    otherdir \
    : Mp \
    (otherdir, Ui \
    G Simp1 \
    casehyp2)

line49
 : [(otherdir_1
    : that
    G E B) =>
    (---
    : that
    G E D2)]

{move
 11}

>>> open

   {move
    13}

   >>> \
       declare \
       eqhyp2 \
       that \
       G E Usc \
       thelaw \
       D2

   eqhyp2
    : that
    G E Usc
    (thelaw
```

```
 (D2))

{move
 13}

>>> \
    define \
    eqhypa2 \
    eqhyp2 \
    : Oridem \
    (Iff1 \
    (eqhyp2, Ui \
    G, Pair \
    (thelaw \
    D2, thelaw \
    D2)))

eqhypa2
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    G = thelaw
    (D2))]

{move
 12}

>>> \
    define \
    line50 \
    eqhyp2 \
    : Subs1 \
    eqhypa2 \
    eqhyp2 \
```

```
    otherdir

line50
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    thelaw
    (D2) E B)]

{move
 12}

>>> \
    open

   {move
    14}

   >>> \
       declare \
       impossiblesub \
       that \
       B <<= \
       prime \
       D2

   impossiblesub
    : that
    B <<=
    prime
    (D2)

   {move
    14}
```

```
>>> \
    define \
    line51 \
    impossiblesub \
    : Mp \
    (line50 \
    eqhyp2, Ui \
    (thelaw \
    D2, Simp1 \
    impossiblesub))

line51
 : [(impossiblesub_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    thelaw
    (D2) E prime
    (D2))]

{move
 13}

>>> \
    define \
    line52 \
    impossiblesub \
    : Iff1 \
    (line51 \
    impossiblesub, Ui \
    thelaw \
    D2, Separation4 \
    Refleq \
    prime \
```

```
    D2)

line52
 : [(impossiblesub_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
    (thelaw
    (D2) E D2) & ~ (thelaw
    (D2) E Usc
    (thelaw
    (D2))))]

{move
 13}

>>> \
    define \
    line53 \
    impossiblesub \
    : Mp \
    (Inusc2 \
    thelaw \
    D2, Simp2 \
    line52 \
    impossiblesub)

line53
 : [(impossiblesub_1
    : that
    B <<=
    prime
    (D2)) =>
    (---
    : that
```

```
        ??)]

    {move
     13}

    >>> \
        close

{move
 13}

>>> \
    define \
    line54 \
    eqhyp2 \
    : Negintro \
    line53

line54
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    ~ (B <<=
    prime
    (D2)))]

{move
 12}

>>> \
    define \
    line55 \
    eqhyp2 \
    : Ds1 \
```

```
    line29 \
    dhyp \
    line54 \
    eqhyp2

line55
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    prime
    (D2) <<=
    B)]

{move
 12}

>>> \
    open

   {move
    14}

   >>> \
       declare \
       H obj

   H : obj

   {move
    14}

   >>> \
       open
```

```
{move
 15}

>>> \
    declare \
    hhyp \
    that \
    H E D2

hhyp
 : that
 H E D2

{move
 15}

>>> \
    define \
    line56 \
    : Excmid \
    (H = thelaw \
    D2)

line56
 : that
 (H = thelaw
 (D2)) V ~ (H = thelaw
 (D2))

{move
 14}

>>> \
    open

   {move
    16}
```

```
>>> \
    declare \
    casehhyp1 \
    that \
    H = thelaw \
    D2

casehhyp1
 : that
 H = thelaw
 (D2)

{move
 16}

>>> \
    declare \
    casehhyp2 \
    that \
    ~ (H = thelaw \
    D2)

casehhyp2
 : that
 ~ (H = thelaw
 (D2))

{move
 16}

>>> \
    define \
    line57 \
    casehhyp1 \
    : Subs1 \
    (Eqsymm \
    casehhyp1, line50 \
    eqhyp2)
```

```
line57
 : [(casehhyp1_1
    : that
    H = thelaw
    (D2)) =>
    (---
    : that
    H E B)]

{move
 15}

>>> \
    open

   {move
    17}

   >>> \
       declare \
       sillyhyp \
       that \
       H E Usc \
       thelaw \
       D2

   sillyhyp
    : that
    H E Usc
    (thelaw
    (D2))

   {move
    17}

   >>> \
       define \
```

```
        line58 \
        sillyhyp \
        : Mp \
        (Oridem \
        (Iff1 \
        (sillyhyp, Ui \
        H, Pair \
        (thelaw \
        D2, thelaw \
        D2))), casehhyp2)

    line58
     : [(sillyhyp_1
        : that
        H E Usc
        (thelaw
        (D2))) =>
        (---
        : that
        ??)]

    {move
     16}

    >>> \
        close

{move
 16}

>>> \
    define \
    line59 \
    casehhyp2 \
    : Negintro \
    line58

line59
```

```
 : [(casehhyp2_1
    : that
    ~ (H = thelaw
    (D2))) =>
    (---
    : that
    ~ (H E Usc
    (thelaw
    (D2))))]

{move
 15}

>>> \
    define \
    line60 \
    casehhyp2 \
    : Fixform \
    (H E prime \
    D2, Iff2 \
    (Conj \
    (hhyp, line59 \
    casehhyp2), Ui \
    H, Separation4 \
    Refleq \
    prime \
    D2))

line60
 : [(casehhyp2_1
    : that
    ~ (H = thelaw
    (D2))) =>
    (---
    : that
    H E prime
    (D2))]
```

```
   {move
    15}

   >>> \
       define \
       line61 \
       casehhyp2 \
       : Mp \
       (line60 \
       casehhyp2, Ui \
       H, Simp1 \
       line55 \
       eqhyp2)

   line61
    : [(casehhyp2_1
       : that
       ~ (H = thelaw
       (D2))) =>
       (---
       : that
       H E B)]

   {move
    15}

   >>> \
       close

{move
 15}

>>> \
    define \
    line62 \
    hhyp \
    : Cases \
    line56 \
```

```
            line57, line61

        line62
         : [(hhyp_1
            : that
            H E D2) =>
            (---
            : that
            H E B)]

        {move
         14}

        >>> \
            close

    {move
     14}

    >>> \
        define \
        line63 \
        H : Ded \
        line62

    line63
     : [(H_1
        : obj) =>
        (---
        : that
        (H_1
        E D2) ->
        H_1
        E B)]

    {move
     13}
```

```
   >>> \
       close

{move
 13}

>>> \
    define \
    line64 \
    eqhyp2 \
    : Ug \
    line63

line64
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    Forall
    ([(x'_2
       : obj) =>
       ({def} (x'_2
       E D2) ->
       x'_2
       E B : prop)]))]

{move
 12}

>>> \
    define \
    line65 \
    eqhyp2 \
    : Fixform \
    (D2 \
```

```
    <<= \
    B, Conj \
    (line64 \
    eqhyp2, Conj \
    (Simp2 \
    Simp2 \
    casehyp2, linea14 \
    bhyp)))

line65
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
    (---
    : that
    D2
    <<=
    B)]

{move
 12}

>>> \
    define \
    line66 \
    eqhyp2 \
    : Antisymsub \
    (casehyp2, line65 \
    eqhyp2)

line66
 : [(eqhyp2_1
    : that
    G E Usc
    (thelaw
    (D2))) =>
```

```
        (---
        : that
        B = D2)]

    {move
     12}

    >>> \
        define \
        line67 \
        eqhyp2 \
        : Mp \
        (Refleq \
        thelaw \
        D2, Subs1 \
        (line66 \
        eqhyp2, casehypa2))

    line67
     : [(eqhyp2_1
        : that
        G E Usc
        (thelaw
        (D2))) =>
        (---
        : that
        ??)]

    {move
     12}

    >>> \
        close

{move
 12}

>>> define \
```

```
        line68 \
        otherdir \
        : Fixform \
        (G E prime \
        D2, Iff2 \
        (Conj \
        (line49 \
        otherdir, Negintro \
        line67), Ui \
        G, Separation4 \
        Refleq \
        prime \
        D2))

    line68
     : [(otherdir_1
        : that
        G E B) =>
        (---
        : that
        G E prime
        (D2))]

    {move
     11}

    >>> close

 {move 11}

 >>> define \
     line69 G : Ded \
     line68

 line69 : [(G_1
     : obj) =>
     (---
     : that
```

```
        (G_1
        E B) ->
        G_1 E prime
        (D2))]

    {move 10}

    >>> define \
        testline \
        G : Ded \
        linea48

    testline
     : [(G_1
        : obj) =>
        (---
        : that
        (G_1
        E prime
        (D2)) ->
        G_1 E prime
        (B))]

    {move 10}

    >>> close

{move 10}

>>> define \
    line70 casehypa2 \
    : Ug line69

line70 : [(casehypa2_1
    : that ~ (thelaw
    (D2) = thelaw
    (B))) =>
    (--- : that
```

```
    Forall ([(x'_2
       : obj) =>
       ({def} (x'_2
       E B) ->
       x'_2
       E prime
       (D2) : prop)]))]

{move 9}

>>> define \
    line71 casehypa2 \
    : Add2 ((prime \
    D2) <<= prime \
    B, Fixform \
    (B <<= prime \
    D2, Conj (line70 \
    casehypa2, Conj \
    (linea14 bhyp, Separation3 \
    Refleq prime \
    D2))))

line71 : [(casehypa2_1
    : that ~ (thelaw
    (D2) = thelaw
    (B))) =>
    (--- : that
    (prime
    (D2) <<=
    prime (B)) V B <<=
    prime (D2))]

{move 9}

>>> define \
    testline2 casehypa1 \
    : Ug testline
```

```
testline2 : [(casehypa1_1
    : that thelaw
    (D2) = thelaw
    (B)) =>
    (--- : that
    Forall ([(x'_2
       : obj) =>
       ({def} (x'_2
       E prime
       (D2)) ->
       x'_2
       E prime
       (B) : prop)]))]

{move 9}

>>> define \
    line72 casehypa1 \
    : Add1 (B <<= \
    prime D2, Fixform \
    ((prime D2) <<= \
    prime B, Conj \
    (testline2 \
    casehypa1, Conj \
    (Separation3 \
    Refleq prime \
    D2, Separation3 \
    Refleq prime \
    B))))

line72 : [(casehypa1_1
    : that thelaw
    (D2) = thelaw
    (B)) =>
    (--- : that
    (prime
    (D2) <<=
    prime (B)) V B <<=
```

```
        prime (D2))]

      {move 9}

      >>> close

   {move 9}

   >>> define line73 \
       casehyp2 : Cases \
       line31 line72, line71

   line73 : [(casehyp2_1
       : that B <<=
       D2) => (---
       : that (prime
       (D2) <<=
       prime (B)) V B <<=
       prime (D2))]

   {move 8}

   >>> close

{move 8}

>>> define line74 \
    Casehyp2 : Cases \
    (line25 dhyp, linea30, line73)

line74 : [(Casehyp2_1
    : that Exists
    ([(U_3 : obj) =>
       ({def} U_3
       E B : prop)])) =>
    (--- : that (prime
    (D2) <<= prime
    (B)) V B <<=
```

```
      prime (D2))]

   {move 7}

   >>> close

{move 7}

>>> define line75 dhyp \
    : Cases (linea14 bhyp, linea29, line74)

line75 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that (prime
    (D2) <<= prime
    (B)) V B <<= D2
    Set [(x_4 : obj) =>
       ({def} ~ (x_4
       E Usc (thelaw
       (D2))) : prop)])]

{move 6}

>>> define line76 dhyp \
    : Fixform ((prime \
    D2) E Cuts2, Iff2 \
    (Conj (line28 dhyp, line75 \
    dhyp), Ui prime D2, Separation4 \
    Refleq Cuts2))

line76 : [(dhyp_1
    : that D2 E Cuts2) =>
    (--- : that prime
    (D2) E Cuts2)]

{move 6}

>>> close
```

```
      {move 6}

      >>> define line77 D2 : Ded \
          line76

      line77 : [(D2_1 : obj) =>
          (--- : that (D2_1
          E Cuts2) -> prime (D2_1) E Cuts2)]

      {move 5}

      >>> close

   {move 5}

   >>> define linea78 : Ug line77

   linea78 : that Forall ([(x'_2
       : obj) =>
       ({def} (x'_2 E Cuts2) ->
       prime (x'_2) E Cuts2
       : prop)])

   {move 4}

   >>> save

   {move 5}

   >>> close

{move 4}

>>> define lineb78 bhyp : linea78

lineb78 : [(bhyp_1 : that B E Cuts) =>
    (--- : that Forall ([(x'_2
```

```
          : obj) =>
          ({def} (x'_2 E Mbold
          Set [(Y_5 : obj) =>
             ({def} cutsh2 (Y_5) : prop)]) ->
          prime (x'_2) E Mbold
          Set [(Y_5 : obj) =>
             ({def} cutsh2 (Y_5) : prop)] : prop)]))]

   {move 3}

   >>> save

   {move 4}

   >>> close

{move 3}

>>> declare bhypa1 that B E Cuts

bhypa1 : that B E Cuts

{move 3}

>>> define linec78 bhypa1 : lineb78 \
    bhypa1

linec78 : [(.B_1 : obj), (bhypa1_1
    : that .B_1 E Cuts) => (---
    : that Forall ([(x'_2 : obj) =>
       ({def} (x'_2 E Mbold Set
       [(Y_5 : obj) =>
          ({def} .B_1 cutsg2 Y_5
          : prop)]) -> prime (x'_2) E Mbold
       Set [(Y_5 : obj) =>
          ({def} .B_1 cutsg2 Y_5
          : prop)] : prop)]))]
```

```
   {move 2}

   >>> save

   {move 3}

   >>> close

{move 2}

>>> declare B111 obj

B111 : obj

{move 2}

>>> declare bhypa2 that B111 E Cuts

bhypa2 : that B111 E Cuts

{move 2}

>>> define lined78 bhypa2 : linec78 \
    bhypa2

lined78 : [(.B111_1 : obj), (bhypa2_1
    : that .B111_1 E Cuts) => (---
    : that Forall ([(x'_2 : obj) =>
       ({def} (x'_2 E Mbold Set [(Y_5
          : obj) =>
          ({def} .B111_1 cutsf2 Y_5
          : prop)]) -> prime (x'_2) E Mbold
       Set [(Y_5 : obj) =>
          ({def} .B111_1 cutsf2 Y_5
          : prop)] : prop)]))]

{move 1}
```

```
   >>> save

   {move 2}

   >>> close

{move 1}

>>> declare B112 obj

B112 : obj

{move 1}

>>> declare bhypa3 that B112 E Cuts

bhypa3 : that B112 E Cuts

{move 1}

>>> define linee78 Misset, thelawchooses, bhypa3 \
    : lined78 bhypa3

linee78 : [(.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)]), (.B112_1
    : obj), (bhypa3_1 : that .B112_1
    E Misset_1 Cuts3 thelawchooses_1) =>
    ({def} Ug ([(D2_2 : obj) =>
       ({def} Ded ([(dhyp_3 : that
          D2_2 E Misset_1 Mbold2 thelawchooses_1
          Set [(Y_6 : obj) =>
             ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_6) : prop)
```

```
({def} (prime2 (.thelaw_1, D2_2) E Misset_1
Mbold2 thelawchooses_1 Set [(Y_6
   : obj) =>
   ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_6) : prop)
Simp1 (dhyp_3 Iff1 D2_2 Ui Separation4
(Refleq (Misset_1 Mbold2 thelawchooses_1
Set [(Y_13 : obj) =>
   ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_13) : prop
D2_2 Ui Simp1 (Simp2 (Simp2
(Misset_1 Mboldtheta2 thelawchooses_1))) Conj
Cases (Setsinchains2 (Misset_1, thelawchooses_1, Misset_1
Mboldtheta2 thelawchooses_1, Simp1
(bhypa3_1 Iff1 .B112_1 Ui Misset_1
Mbold2 thelawchooses_1 Separation
[(C_12 : obj) =>
   ({def} cuts2 (Misset_1, thelawchooses_1, C_12) : prop)])), [(Ca
   : that .B112_1 = 0) =>
   ({def} Eqsymm (Casehyp1_7) Subs1
   (prime2 (.thelaw_1, D2_2) <<=
   prime2 (.thelaw_1, .B112_1)) Add2
   Zeroissubset (Separation3
   (Refleq (prime2 (.thelaw_1, D2_2)))) : that
   (prime2 (.thelaw_1, D2_2) <<=
   prime2 (.thelaw_1, .B112_1)) V .B112_1
   <<= D2_2 Set [(x_10 : obj) =>
      ({def} ~ (x_10 E Usc
      (.thelaw_1 (D2_2))) : prop)])], [(Casehyp2_7
   : that Exists ([(U_9 : obj) =>
      ({def} U_9 E .B112_1 : prop)])) =>
   ({def} Cases (Simp2 (dhyp_3
   Iff1 D2_2 Ui Separation4 (Refleq
   (Misset_1 Mbold2 thelawchooses_1
   Set [(Y_14 : obj) =>
      ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_14) : p
      : that D2_2 <<= prime2
      (.thelaw_1, .B112_1)) =>
      ({def} (.B112_1 <<= prime2
      (.thelaw_1, D2_2)) Add1
```

```
((prime2 (.thelaw_1, D2_2) <<=
D2_2) Fixform Setsinchains2
(Misset_1, thelawchooses_1, Misset_1
Mboldtheta2 thelawchooses_1, Simp1
(dhyp_3 Iff1 D2_2 Ui Separation4
(Refleq (Misset_1 Mbold2
thelawchooses_1 Set [(Y_19
   : obj) =>
   ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_19)
Refleq (prime2 (.thelaw_1, D2_2))) Transsub
casehyp1_8 : that (prime2
(.thelaw_1, D2_2) <<=
prime2 (.thelaw_1, .B112_1)) V .B112_1
<<= prime2 (.thelaw_1, D2_2))], [(casehyp2_8
: that .B112_1 <<= D2_2) =>
({def} Cases (Excmid
(.thelaw_1 (D2_2) = .thelaw_1
(.B112_1)), [(casehypa1_9
   : that .thelaw_1 (D2_2) = .thelaw_1
   (.B112_1)) =>
   ({def} (.B112_1 <<=
   prime2 (.thelaw_1, D2_2)) Add1
   (prime2 (.thelaw_1, D2_2) <<=
   prime2 (.thelaw_1, .B112_1)) Fixform
   Ug ([(G_13 : obj) =>
      ({def} Ded ([(onedir_14
         : that G_13 E prime2
         (.thelaw_1, D2_2)) =>
         ({def} (G_13
         E prime2 (.thelaw_1, .B112_1)) Fixform
         Cases (Simp1
         (dhyp_3 Iff1
         D2_2 Ui Separation4
         (Refleq (Misset_1
         Mbold2 thelawchooses_1
         Set [(Y_26 : obj) =>
            ({def} cutse2
            (Misset_1, thelawchooses_1, .B112_1, Y_26) : pro
```

```
D2_2 Ui Simp1
(Simp2 (Simp2
(Misset_1 Mboldtheta2
thelawchooses_1))) Mp
prime2 (.thelaw_1, D2_2) Ui
Simp2 (Simp2
(bhypa3_1 Iff1
.B112_1 Ui Separation4
(Refleq (Misset_1
Cuts3 thelawchooses_1)))), [(casehypb1_18
   : that prime2
   (.thelaw_1, D2_2) <<=
   .B112_1) =>
   ({def} onedir_14
   Mp G_13 Ui
   Simp1 (casehypb1_18) : that
   G_13 E .B112_1)], [(casehypb2_18
   : that .B112_1
   <<= prime2
   (.thelaw_1, D2_2)) =>
   ({def} (G_13
   E .B112_1) Giveup
   Inusc2 (.thelaw_1
   (D2_2)) Mp
   Simp2 (Eqsymm
   (casehypa1_9) Subs1
   thelawchooses_1
   (.B112_1, Simp1
   (bhypa3_1
   Iff1 .B112_1
   Ui Misset_1
   Mbold2 thelawchooses_1
   Separation
   [(C_31 : obj) =>
      ({def} cuts2
      (Misset_1, thelawchooses_1, C_31) : prop)]) M
   .B112_1 Ui
   Simp1 (Simp1
```

```
          (Simp2 (Misset_1
          Mboldtheta2
          thelawchooses_1))) Iff1
          .B112_1 Ui
          Scthm (.M_1), Casehyp2_7) Mp
          .thelaw_1 (.B112_1) Ui
          Simp1 (casehypb2_18) Iff1
          .thelaw_1 (.B112_1) Ui
          Separation4
          (Refleq (prime2
          (.thelaw_1, D2_2)))) : that
          G_13 E .B112_1)]) Conj
       casehypa1_9 Subs1
       Simp2 (onedir_14
       Iff1 G_13 Ui Separation4
       (Refleq (prime2
       (.thelaw_1, D2_2)))) Iff2
       G_13 Ui Separation4
       (Refleq (prime2
       (.thelaw_1, .B112_1))) : that
       G_13 E prime2
       (.thelaw_1, .B112_1))]) : that
   (G_13 E prime2 (.thelaw_1, D2_2)) ->
   G_13 E prime2 (.thelaw_1, .B112_1))]) Conj
Separation3 (Refleq
(prime2 (.thelaw_1, D2_2))) Conj
Separation3 (Refleq
(prime2 (.thelaw_1, .B112_1))) : that
(prime2 (.thelaw_1, D2_2) <<=
prime2 (.thelaw_1, .B112_1)) V .B112_1
<<= prime2 (.thelaw_1, D2_2))], [(casehypa2_9
: that ~ (.thelaw_1
(D2_2) = .thelaw_1
(.B112_1))) =>
({def} (prime2 (.thelaw_1, D2_2) <<=
prime2 (.thelaw_1, .B112_1)) Add2
(.B112_1 <<= prime2
(.thelaw_1, D2_2)) Fixform
```

```
Ug ([(G_13 : obj) =>
   ({def} Ded ([(otherdir_14
      : that G_13 E .B112_1) =>
      ({def} (G_13
      E prime2 (.thelaw_1, D2_2)) Fixform
      otherdir_14 Mp
      G_13 Ui Simp1
      (casehyp2_8) Conj
      Negintro ([(eqhyp2_18
         : that G_13
         E Usc (.thelaw_1
         (D2_2))) =>
         ({def} Refleq
         (.thelaw_1
         (D2_2)) Mp
         casehyp2_8
         Antisymsub
         (D2_2 <<=
         .B112_1) Fixform
         Ug ([(H_24
            : obj) =>
            ({def} Ded
            ([(hhyp_25
               : that
               H_24
               E D2_2) =>
               ({def} Cases
               (Excmid
               (H_24
               = .thelaw_1
               (D2_2)), [(casehhyp1_26
                  : that
                  H_24
                  = .thelaw_1
                  (D2_2)) =>
                  ({def} Eqsymm
                  (casehhyp1_26) Subs1
                  Oridem
```

```
(eqhyp2_18
Iff1
G_13
Ui
.thelaw_1
(D2_2) Pair
.thelaw_1
(D2_2)) Subs1
otherdir_14
: that
H_24
E .B112_1)], [(casehhyp2_26
: that
~ (H_24
= .thelaw_1
(D2_2))) =>
({def} ((H_24
E prime2
(.thelaw_1, D2_2)) Fixform
hhyp_25
Conj
Negintro
([(sillyhyp_31
   : that
   H_24
   E Usc
   (.thelaw_1
   (D2_2))) =>
   ({def} Oridem
   (sillyhyp_31
   Iff1
   H_24
   Ui
   .thelaw_1
   (D2_2) Pair
   .thelaw_1
   (D2_2)) Mp
   casehhyp2_26
```

```
   : that
   ??)]) Iff2
H_24
Ui
Separation4
(Refleq
(prime2
(.thelaw_1, D2_2)))) Mp
H_24
Ui
Simp1
(Simp1
(dhyp_3
Iff1
D2_2
Ui
Separation4
(Refleq
(Misset_1
Mbold2
thelawchooses_1
Set
[(Y_38
   : obj) =>
   ({def} cutse2
   (Misset_1, thelawchooses_1, .B112_1,
D2_2
Ui
Simp1
(Simp2
(Simp2
(Misset_1
Mboldtheta2
thelawchooses_1))) Mp
prime2
(.thelaw_1, D2_2) Ui
Simp2
(Simp2
```

```
(bhypa3_1
Iff1
.B112_1
Ui
Separation4
(Refleq
(Misset_1
Cuts3
thelawchooses_1)))) Ds1
Negintro
([(impossiblesub_31
   : that
   .B112_1
   <<=
   prime2
   (.thelaw_1, D2_2)) =>
   ({def} Inusc2
   (.thelaw_1
   (D2_2)) Mp
   Simp2
   (Oridem
   (eqhyp2_18
   Iff1
   G_13
   Ui
   .thelaw_1
   (D2_2) Pair
   .thelaw_1
   (D2_2)) Subs1
   otherdir_14
   Mp
   .thelaw_1
   (D2_2) Ui
   Simp1
   (impossiblesub_31) Iff1
   .thelaw_1
   (D2_2) Ui
   Separation4
```

```
                        (Refleq
                        (prime2
                        (.thelaw_1, D2_2)))) : that
                        ??)])) : that
                     H_24
                     E .B112_1)]) : that
                  H_24
                  E .B112_1)]) : that
               (H_24 E D2_2) ->
               H_24 E .B112_1)]) Conj
            Simp2 (Simp2
            (casehyp2_8)) Conj
            Setsinchains2
            (Misset_1, thelawchooses_1, Misset_1
            Mboldtheta2
            thelawchooses_1, Simp1
            (bhypa3_1
            Iff1 .B112_1
            Ui Misset_1
            Mbold2 thelawchooses_1
            Separation
            [(C_29 : obj) =>
               ({def} cuts2
               (Misset_1, thelawchooses_1, C_29) : prop)]))
            casehypa2_9
            : that ??)]) Iff2
         G_13 Ui Separation4
         (Refleq (prime2
         (.thelaw_1, D2_2))) : that
         G_13 E prime2
         (.thelaw_1, D2_2))]) : that
      (G_13 E .B112_1) ->
      G_13 E prime2 (.thelaw_1, D2_2))]) Conj
   Setsinchains2 (Misset_1, thelawchooses_1, Misset_1
   Mboldtheta2 thelawchooses_1, Simp1
   (bhypa3_1 Iff1 .B112_1
   Ui Misset_1 Mbold2 thelawchooses_1
   Separation [(C_18
```

```
                    : obj) =>
                    ({def} cuts2 (Misset_1, thelawchooses_1, C_18) : prop)
                 Separation3 (Refleq
                 (prime2 (.thelaw_1, D2_2))) : that
                 (prime2 (.thelaw_1, D2_2) <<=
                 prime2 (.thelaw_1, .B112_1)) V .B112_1
                 <<= prime2 (.thelaw_1, D2_2))]) : that
              (prime2 (.thelaw_1, D2_2) <<=
              prime2 (.thelaw_1, .B112_1)) V .B112_1
              <<= prime2 (.thelaw_1, D2_2))]) : that
           (prime2 (.thelaw_1, D2_2) <<=
           prime2 (.thelaw_1, .B112_1)) V .B112_1
           <<= prime2 (.thelaw_1, D2_2))]) Iff2
        prime2 (.thelaw_1, D2_2) Ui
        Separation4 (Refleq (Misset_1
        Mbold2 thelawchooses_1 Set [(Y_9
           : obj) =>
           ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_9) : prop)
        prime2 (.thelaw_1, D2_2) E Misset_1
        Mbold2 thelawchooses_1 Set [(Y_5
           : obj) =>
           ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)
     (D2_2 E Misset_1 Mbold2 thelawchooses_1
     Set [(Y_5 : obj) =>
        ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)])
     prime2 (.thelaw_1, D2_2) E Misset_1
     Mbold2 thelawchooses_1 Set [(Y_5
        : obj) =>
        ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)])]
  Forall ([(x'_2 : obj) =>
     ({def} (x'_2 E Misset_1 Mbold2
     thelawchooses_1 Set [(Y_5 : obj) =>
        ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)])
     prime2 (.thelaw_1, x'_2) E Misset_1
     Mbold2 thelawchooses_1 Set [(Y_5
        : obj) =>
        ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)] :
```

```
linee78 : [(.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)]), (.B112_1
    : obj), (bhypa3_1 : that .B112_1
    E Misset_1 Cuts3 thelawchooses_1) =>
    (--- : that Forall ([(x'_2 : obj) =>
       ({def} (x'_2 E Misset_1 Mbold2
       thelawchooses_1 Set [(Y_5 : obj) =>
          ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)])
       prime2 (.thelaw_1, x'_2) E Misset_1
       Mbold2 thelawchooses_1 Set [(Y_5
          : obj) =>
          ({def} cutse2 (Misset_1, thelawchooses_1, .B112_1, Y_5) : prop)] :

{move 0}

>>> open

   {move 2}

   >>> define linead78 bhypa2 : linee78 \
       Misset, thelawchooses, bhypa2

   linead78 : [(.B111_1 : obj), (bhypa2_1
       : that .B111_1 E Cuts) => (---
       : that Forall ([(x'_2 : obj) =>
          ({def} (x'_2 E Misset Mbold2
          thelawchooses Set [(Y_5 : obj) =>
             ({def} cutse2 (Misset, thelawchooses, .B111_1, Y_5) : prop)]) -
          prime2 ([(S'_5 : obj) =>
             ({def} thelaw (S'_5) : obj)], x'_2) E Misset
          Mbold2 thelawchooses Set [(Y_5
             : obj) =>
```

```
        ({def} cutse2 (Misset, thelawchooses, .B111_1, Y_5) : prop)] :

{move 1}

>>> open

   {move 3}

   >>> define lineac78 bhypa1 : linead78 \
       bhypa1

   lineac78 : [(.B_1 : obj), (bhypa1_1
       : that .B_1 E Cuts) => (---
       : that Forall ([(x'_2 : obj) =>
          ({def} (x'_2 E Misset Mbold2
          thelawchooses Set [(Y_5
             : obj) =>
             ({def} cutse2 (Misset, thelawchooses, .B_1, Y_5) : prop)]) -
          prime2 ([(S'_5 : obj) =>
             ({def} thelaw (S'_5) : obj)], x'_2) E Misset
          Mbold2 thelawchooses Set [(Y_5
             : obj) =>
             ({def} cutse2 (Misset, thelawchooses, .B_1, Y_5) : prop)] :

   {move 2}

   >>> open

      {move 4}

      >>> define lineab78 bhyp : lineac78 \
          bhyp

      lineab78 : [(bhyp_1 : that
          B E Cuts) => (--- : that
          Forall ([(x'_2 : obj) =>
             ({def} (x'_2 E Misset
             Mbold2 thelawchooses Set
```

```
            [(Y_5 : obj) =>
               ({def} cutse2 (Misset, thelawchooses, B, Y_5) : prop)]) -
            prime2 ([(S'_5 : obj) =>
               ({def} thelaw (S'_5) : obj)], x'_2) E Misset
            Mbold2 thelawchooses Set
            [(Y_5 : obj) =>
               ({def} cutse2 (Misset, thelawchooses, B, Y_5) : prop)] :

      {move 3}

      >>> open

         {move 5}

         >>> define line78 : lineab78 \
             bhyp

         line78 : that Forall ([(x'_2
             : obj) =>
             ({def} (x'_2 E Misset
             Mbold2 thelawchooses Set
             [(Y_5 : obj) =>
                ({def} cutse2 (Misset, thelawchooses, B, Y_5) : prop)]) -
             prime2 ([(S'_5 : obj) =>
                ({def} thelaw (S'_5) : obj)], x'_2) E Misset
             Mbold2 thelawchooses Set
             [(Y_5 : obj) =>
                ({def} cutse2 (Misset, thelawchooses, B, Y_5) : prop)] :

         {move 4}
end Lestrade execution
```

This is the third component of the proof that Cuts2 is a Θ-chain. I want to examine the proof strategy; I also want to see if the size of the term and the slowness of generation of the term can be improved by exporting some intermediate stages to move 0.

begin Lestrade execution

```
>>> goal that Forall [D1 \
       => Forall [F1 => ((D1 \
          <<= Cuts2) & F1 E D1) -> \
          (D1 Intersection F1) E Cuts2]]

that Forall ([(D1 : obj) =>
    ({def} Forall ([(F1
       : obj) =>
       ({def} ((D1 <<= Cuts2) & F1
       E D1) -> (D1 Intersection
       F1) E Cuts2 : prop)]) : prop)])

{move 5}

>>> open

   {move 6}

   >>> declare D2 obj

   D2 : obj

   {move 6}

   >>> open

      {move 7}

      >>> declare F2 obj

      F2 : obj

      {move 7}

      >>> open
```

```
{move 8}

>>> declare intev \
    that (D2 <<= Cuts2) & F2 \
    E D2

intev : that (D2
 <<= Cuts2) & F2
 E D2

{move 8}

>>> goal that (D2 \
    Intersection F2) E Cuts2

that (D2 Intersection
 F2) E Cuts2

{move 8}

>>> define line79 \
    : Ui D2 Intersection \
    F2, Separation4 \
    Refleq Cuts2

line79 : that ((D2
 Intersection F2) E Mbold
 Set cutsi2) == ((D2
 Intersection F2) E Mbold) & cutsi2
 (D2 Intersection
 F2)

{move 7}

>>> goal that (D2 \
    Intersection F2) E Mbold

that (D2 Intersection
```

```
 F2) E Mbold

{move 8}

>>> define line80 \
    : Ui F2, Ui D2, Simp2 \
    (Simp2 (Simp2 Mboldtheta))

line80 : that ((D2
 <<= Misset Mbold2
 thelawchooses) & F2
 E D2) -> (D2 Intersection
 F2) E Misset Mbold2
 thelawchooses

{move 7}

>>> define line81 \
    intev : Mp (Conj \
    (Transsub (Simp1 \
    intev, line20), Simp2 \
    intev), line80)

line81 : [(intev_1
    : that (D2 <<=
    Cuts2) & F2 E D2) =>
    (--- : that (D2
    Intersection F2) E Misset
    Mbold2 thelawchooses)]

{move 7}

>>> goal that ((D2 \
    Intersection F2) <<= \
    prime B) V B <<= \
    D2 Intersection F2

that ((D2 Intersection
```

```
 F2) <<= prime (B)) V B <<=
 D2 Intersection F2

{move 8}

>>> declare K obj

K : obj

{move 8}

>>> define line82 \
    : Excmid Forall [K => \
       (K E D2) -> \
       B <<= K]

line82 : that Forall
 ([(K_3 : obj) =>
    ({def} (K_3
    E D2) -> B <<=
    K_3 : prop)]) V ~ (Forall
 ([(K_4 : obj) =>
    ({def} (K_4
    E D2) -> B <<=
    K_4 : prop)]))

{move 7}

>>> open

   {move 9}

   >>> goal that \
       ((D2 Intersection \
       F2) <<= prime \
       B) V B <<= D2 \
       Intersection F2
```

```
that ((D2 Intersection
 F2) <<= prime
 (B)) V B <<=
 D2 Intersection
 F2

{move 9}

>>> declare K1 \
    obj

K1 : obj

{move 9}

>>> declare casehyp1 \
    that Forall [K1 \
       => (K1 E D2) -> \
       B <<= K1]

casehyp1 : that
 Forall ([(K1_2
    : obj) =>
    ({def} (K1_2
    E D2) -> B <<=
    K1_2 : prop)])

{move 9}

>>> goal that \
    B <<= D2 Intersection \
    F2

that B <<= D2
 Intersection F2

{move 9}
```

```
>>> open

   {move 10}

   >>> declare \
       K2 obj

   K2 : obj

   {move 10}

   >>> open

      {move 11}

      >>> declare \
          khyp that \
          K2 E B

      khyp : that
       K2 E B

      {move 11}

      >>> open

         {move
          12}

         >>> declare \
             B2 obj

         B2 : obj

         {move
          12}

         >>> open
```

```
{move
 13}

>>> \
    declare \
    bhyp2 \
    that \
    B2 \
    E D2

bhyp2
 : that
 B2
 E D2

{move
 13}

>>> \
    define \
    line83 \
    bhyp2 \
    : Mpsubs \
    (khyp, Mp \
    (bhyp2, Ui \
    B2, casehyp1))

line83
 : [(bhyp2_1
    : that
    B2
    E D2) =>
    (---
    : that
    K2
    E B2)]
```

```
        {move
         12}

        >>> \
            close

     {move
      12}

     >>> define \
         line84 \
         B2 : Ded \
         line83

     line84
      : [(B2_1
         : obj) =>
         (---
         : that
         (B2_1
         E D2) ->
         K2
         E B2_1)]

     {move
      11}

     >>> close

  {move 11}

  >>> define \
      line85 khyp \
      : Ug line84

  line85 : [(khyp_1
      : that
      K2 E B) =>
```

```
    (---
    : that
    Forall
    ([(x'_2
       : obj) =>
       ({def} (x'_2
       E D2) ->
       K2
       E x'_2
       : prop)]))]

{move 10}

>>> define \
    line86 khyp \
    : Mp (Simp2 \
    intev, Ui \
    F2, line85 \
    khyp)

line86 : [(khyp_1
    : that
    K2 E B) =>
    (---
    : that
    K2 E F2)]

{move 10}

>>> define \
    line87 khyp \
    : Fixform \
    (K2 E D2 \
    Intersection \
    F2, Iff2 \
    (Conj (line86 \
    khyp, line85 \
    khyp), Ui \
```

```
            K2, Separation4 \
            Refleq (D2 \
            Intersection \
            F2)))

        line87 : [(khyp_1
            : that
            K2 E B) =>
            (---
            : that
            K2 E D2
            Intersection
            F2)]

        {move 10}

        >>> close

    {move 10}

    >>> define \
        line88 K2 : Ded \
        line87

    line88 : [(K2_1
        : obj) =>
        (--- : that
        (K2_1 E B) ->
        K2_1 E D2
        Intersection
        F2)]

    {move 9}

    >>> close

{move 9}
```

```
>>> define line89 \
    casehyp1 : Fixform \
    (B <<= D2 Intersection \
    F2, Conj (Ug \
    line88, Conj \
    (linea14 bhyp, Separation3 \
    Refleq (D2 Intersection \
    F2))))

line89 : [(casehyp1_1
    : that Forall
    ([(K1_3
       : obj) =>
       ({def} (K1_3
       E D2) ->
       B <<= K1_3
       : prop)])) =>
    (--- : that
    B <<= D2 Intersection
    F2)]

{move 8}

>>> define line90 \
    casehyp1 : Add2 \
    ((D2 Intersection \
    F2) <<= prime \
    B, line89 casehyp1)

line90 : [(casehyp1_1
    : that Forall
    ([(K1_3
       : obj) =>
       ({def} (K1_3
       E D2) ->
       B <<= K1_3
       : prop)])) =>
    (--- : that
```

```
    ((D2 Intersection
    F2) <<= prime
    (B)) V B <<=
    D2 Intersection
    F2)]

{move 8}

>>> declare casehyp2 \
    that ~ (Forall \
    [K1 => (K1 E D2) -> \
       B <<= K1])

casehyp2 : that
 ~ (Forall ([(K1_3
    : obj) =>
    ({def} (K1_3
    E D2) -> B <<=
    K1_3 : prop)]))

{move 9}

>>> goal that \
    ((D2 Intersection \
    F2) <<= prime \
    B)

that (D2 Intersection
 F2) <<= prime
 (B)

{move 9}

>>> open

   {move 10}

   >>> declare \
```

```
    K2 obj

K2 : obj

{move 10}

>>> open

   {move 11}

   >>> declare \
       khyp2 that \
       K2 E D2 \
       Intersection \
       F2

   khyp2 : that
    K2 E D2
    Intersection
    F2

   {move 11}

   >>> define \
       line91 : Counterexample \
       casehyp2

   line91 : that
    Exists ([(z_2
       : obj) =>
       ({def} ~ ((z_2
       E D2) ->
       B <<=
       z_2) : prop)])

   {move 10}

   >>> open
```

```
{move
 12}

>>> declare \
    F3 obj

F3 : obj

{move
 12}

>>> declare \
    fhyp3 \
    that \
    Witnesses \
    line91 \
    F3

fhyp3
 : that
 line91
 Witnesses
 F3

{move
 12}

>>> define \
    line92 \
    fhyp3 \
    : Notimp2 \
    fhyp3

line92
 : [(.F3_1
    : obj), (fhyp3_1
    : that
```

```
    line91
    Witnesses
    .F3_1) =>
    (---
    : that
    .F3_1
    E D2)]

{move
 11}

>>> define \
    line93 \
    fhyp3 \
    : Notimp1 \
    fhyp3

line93
 : [(.F3_1
    : obj), (fhyp3_1
    : that
    line91
    Witnesses
    .F3_1) =>
    (---
    : that
    ~ (B <<=
    .F3_1))]

{move
 11}

>>> define \
    line94 \
    fhyp3 \
    : Simp2 \
    (Iff1 \
    (Mpsubs \
```

```
    (line92 \
    fhyp3, Simp1 \
    intev), Ui \
    F3, Separation4 \
    Refleq \
    Cuts2))

line94
 : [(.F3_1
    : obj), (fhyp3_1
    : that
    line91
    Witnesses
    .F3_1) =>
    (---
    : that
    cutsi2
    (.F3_1))]

{move
 11}

>>> define \
    line95 \
    fhyp3 \
    : Ds1 \
    (line94 \
    fhyp3, line93 \
    fhyp3)

line95
 : [(.F3_1
    : obj), (fhyp3_1
    : that
    line91
    Witnesses
    .F3_1) =>
    (---
```

```
    : that
    .F3_1
    <<=
    prime2
    ([(S'_3
       : obj) =>
       ({def} thelaw
       (S'_3) : obj)], B))]

{move
 11}

>>> define \
    line96 \
    fhyp3 \
    : Mp \
    line92 \
    fhyp3, Ui \
    F3, Simp2 \
    (Iff1 \
    khyp2, Ui \
    K2, Separation4 \
    Refleq \
    (D2 \
    Intersection \
    F2))

line96
 : [(.F3_1
    : obj), (fhyp3_1
    : that
    line91
    Witnesses
    .F3_1) =>
    (---
    : that
    K2
    E .F3_1)]
```

```
{move
 11}

>>> define \
    line97 \
    fhyp3 \
    : Mpsubs \
    line96 \
    fhyp3 \
    line95 \
    fhyp3

line97
 : [(.F3_1
    : obj), (fhyp3_1
    : that
    line91
    Witnesses
    .F3_1) =>
    (---
    : that
    K2
    E prime2
    ([(S'_3
       : obj) =>
       ({def} thelaw
       (S'_3) : obj)], B))]

{move
 11}

>>> close

{move 11}

>>> define \
    line98 khyp2 \
```

```
        : Eg line91 \
        line97

    line98 : [(khyp2_1
        : that
        K2 E D2
        Intersection
        F2) =>
        (---
        : that
        K2 E prime2
        ([(S'_3
           : obj) =>
           ({def} thelaw
           (S'_3) : obj)], B))]

    {move 10}

    >>> close

{move 10}

>>> define \
    line99 K2 : Ded \
    line98

line99 : [(K2_1
    : obj) =>
    (--- : that
    (K2_1 E D2
    Intersection
    F2) ->
    K2_1 E prime2
    ([(S'_4
       : obj) =>
       ({def} thelaw
       (S'_4) : obj)], B))]
```

```
   {move 9}

   >>> close

{move 9}

>>> define linea10 \
    casehyp2 : Fixform \
    ((D2 Intersection \
    F2) <<= prime \
    B, Conj (Ug \
    line99, Conj \
    (Separation3 \
    Refleq (D2 Intersection \
    F2), Separation3 \
    Refleq (prime \
    B))))

linea10 : [(casehyp2_1
    : that ~ (Forall
    ([(K1_4
       : obj) =>
       ({def} (K1_4
       E D2) ->
       B <<= K1_4
       : prop)]))) =>
    (--- : that
    (D2 Intersection
    F2) <<= prime
    (B))]

{move 8}

>>> define linea11 \
    casehyp2 : Add1 \
    (B <<= D2 Intersection \
    F2, linea10 casehyp2)
```

```
   linea11 : [(casehyp2_1
       : that ~ (Forall
       ([(K1_4
          : obj) =>
          ({def} (K1_4
          E D2) ->
          B <<= K1_4
          : prop)]))) =>
       (--- : that
       ((D2 Intersection
       F2) <<= prime
       (B)) V B <<=
       D2 Intersection
       F2)]

   {move 8}

   >>> close

{move 8}

>>> define line12 \
    intev : Cases line82 \
    line90, linea11

line12 : [(intev_1
    : that (D2 <<=
    Cuts2) & F2 E D2) =>
    (--- : that ((D2
    Intersection F2) <<=
    prime (B)) V B <<=
    D2 Intersection
    F2)]

{move 7}

>>> define linea12 \
    intev : Conj (line81 \
```

```
        intev, line12 intev)

   linea12 : [(intev_1
        : that (D2 <<=
        Cuts2) & F2 E D2) =>
        (--- : that ((D2
        Intersection F2) E Misset
        Mbold2 thelawchooses) & ((D2
        Intersection F2) <<=
        prime (B)) V B <<=
        D2 Intersection
        F2)]

   {move 7}

   >>> define lineb12 \
        intev : Fixform ((D2 \
        Intersection F2) E Cuts2, Iff2 \
        (linea12 intev, Ui \
        (D2 Intersection \
        F2, Separation4 \
        Refleq Cuts2)))

   lineb12 : [(intev_1
        : that (D2 <<=
        Cuts2) & F2 E D2) =>
        (--- : that (D2
        Intersection F2) E Cuts2)]

   {move 7}

   >>> close

{move 7}

>>> define linea13 F2 \
    : Ded lineb12
```

```
                     linea13 : [(F2_1 : obj) =>
                         (--- : that ((D2
                         <<= Cuts2) & F2_1
                         E D2) -> (D2 Intersection
                         F2_1) E Cuts2)]

                     {move 6}

                     >>> close

                  {move 6}
end Lestrade execution
```