Ωmega

Christoph Benzmüller, Armin Fiedler, Andreas Meier, Martin Pollet, and Jörg Siekmann

Formalization[1] and answers by Christoph Benzmüller <chris@ags.uni-sb.de>, Armin Fiedler <afiedler@ags.uni-sb.de>, Andreas Meier <ameier@ags.uni-sb.de>, Martin Pollet <pollet@ags.uni-sb.de> and Jörg Siekmann <siekmann@ags.uni-sb.de>.

15.1 Statement

```
(not (rat (sqrt 2)))
```

15.2 Definitions

Definition of `rat`

```
(th~defdef rat
        (in rational)
        (definition
          (lam (x num)
               (exists-sort (lam (y num)
                (exists-sort (lam (z num)
                 (= x (frac y z))) int\0)) int)))
        (sort)
        (help "The set of rationals, constructed as fractions a/b of integers."))
```

Definition of `sqrt`

```
(th~defdef sqrt
        (in real)
        (definition
          (lam (x num) (that (lam (y num) (= (power y 2) x)))))
        (help "Definition of square root."))
```

15.3 Proof

Some definitions from the Ωmega library

```
(th~deftheorem rat-criterion
        (in real)
        (conclusion
         (forall-sort (lam (x num)
           (exists-sort (lam (y num)
            (exists-sort (lam (z num)
             (and (= (times x y) z)
```

[1] This paper represents the work as submitted in 2002. Publication has been delayed and in the meantime we have better and far more advanced results (see Section 15.4) on this problem.

F. Wiedijk (Ed.): The Seventeen Provers of the World, LNAI 3600, pp. 127–141, 2006.

```
                  (not (exists-sort (lam (d num)
                         (common-divisor y z d))
                         int))))
             int))
            int))
          rat))
        (help "x rational implies there exist integers y, z which have no common divisor
               and furthermore z = x · y."))

(th~deftheorem even-on-integers
        (in real)
        (conclusion
         (forall-sort (lam (x num)
           (equiv (evenp x)
                  (exists-sort (lam (y num)
                    (= x (times 2 y))) int))) int))
        (help "An integer x is even, iff an integer y exists so that x = 2y."))

(th~deftheorem square-even
        (in real)
        (conclusion
         (forall-sort (lam (x num)
          (equiv (evenp (power x 2)) (evenp x)))
          int))
        (help "x is even, iff x^2 is even."))

(th~deftheorem even-common-divisor
        (in real)
        (conclusion
         (forall-sort (lam (x num)
          (forall-sort (lam (y num)
           (implies (and (evenp x) (evenp y))
                    (common-divisor x y 2)))
           int))
          int))
        (help "If x and y are even, then they have '2' as a common divisor."))

(th~deftheorem power-int-closed
        (in real)
        (conclusion
         (forall-sort (lam (x num)
          (forall-sort (lam (y num)
           (int (power y x)))
           int))
          int))
        (termdecl)
        (help "The set of integers is closed under power."))

(th~defproblem sqrt2-not-rat
        (in real)
        (conclusion (not (rat (sqrt 2))))
        (help "√2 is not a rational number."))
```

Proof script for `sqrt2-not-rat`

```
DECLARATION DECLARE ((CONSTANTS (M NUM) (N NUM) (K NUM)))
RULES NOTI default default
MBASE IMPORT-ASS (RAT-CRITERION)
TACTICS FORALLE-SORT default default ((SQRT 2)) default
```

```
TACTICS EXISTSE-SORT default default (N) default
TACTICS ANDE default default default
TACTICS EXISTSE-SORT (L7) default (M) default
TACTICS ANDE* (L8) (NIL)
OMEGA-BASIC LEMMA default ((= (POWER M 2) (TIMES 2 (POWER N 2))))
TACTICS BY-COMPUTATION (L13) ((L11))
OMEGA-BASIC LEMMA (L9) ((EVENP (POWER M 2)))
RULES DEFN-CONTRACT default default default
OMEGA-BASIC LEMMA (L9) ((INT (POWER N 2)))
TACTICS WELLSORTED default default
TACTICS EXISTSI-SORT (L15) ((POWER N 2)) (L13) (L16) default
MBASE IMPORT-ASS (SQUARE-EVEN)
TACTICS ASSERT ((EVENP M)) ((SQUARE-EVEN L10 L14)) (NIL)
RULES DEFN-EXPAND (L17) default default
TACTICS EXISTSE-SORT default default (K) default
TACTICS ANDE (L19) default default
OMEGA-BASIC LEMMA default ((= (POWER N 2) (TIMES 2 (POWER K 2))))
TACTICS BY-COMPUTATION (L23) ((L13 L22))
OMEGA-BASIC LEMMA default ((EVENP (POWER N 2)))
RULES DEFN-CONTRACT default default default
OMEGA-BASIC LEMMA (L20) ((INT (POWER K 2)))
TACTICS WELLSORTED (L26) ((L21))
TACTICS EXISTSI-SORT default ((POWER K 2)) (L23) default default
TACTICS ASSERT ((EVENP N)) ((SQUARE-EVEN L6 L24)) (NIL)
MBASE IMPORT-ASS (EVEN-COMMON-DIVISOR)
OMEGA-BASIC LEMMA (L20) ((INT 2))
TACTICS WELLSORTED (L28) (NIL)
TACTICS ASSERT (FALSE) ((EVEN-COMMON-DIVISOR L10 L6 L12 L17 L27 L28)) (NIL)
RULES WEAKEN default default
```

First ten steps of Emacs session corresponding to the proof script

We load the theory `Real`, in which the problem is defined.

```
OMEGA: load-problems

THEORY-NAME (EXISTING-THEORY) The name of a theory whose problems ar to be loaded: [REAL]real
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
;;; Rules loaded for theory REAL.
;;; Theorems loaded for theory REAL.
;;; Tactics loaded for theory REAL.
;;; Methods loaded for theory REAL.
```

Step: 1. First we load the problem from the Ωmega database and declare some constant symbols which we will later employ.

```
OMEGA: prove

PROOF-PLAN (PROOF-PLAN) A natural deduction proof or a problem: sqrt2-not-rat
```

```
Changing to proof plan SQRT2-NOT-RAT-21

OMEGA: show-pds

               ...

SQRT2-NOT-RAT ()              ! (NOT (RAT (SQRT 2)))                                              OPEN

OMEGA: declare (constants (m num) (n num) (k num))
```

Step: 2. We prove the goal indirectly.

```
OMEGA: noti

NEGATION (NDLINE) A negated line: [SQRT2-NOT-RAT]

FALSITY (NDLINE) A falsity line: [()]
;;;CSM Arbitrary [2]: 0 provers have to be killed

OMEGA: show-pds

L1            (L1)            ! (RAT (SQRT 2))                                                     HYP
               ...
L2            (L1)            ! FALSE                                                             OPEN

SQRT2-NOT-RAT ()              ! (NOT (RAT (SQRT 2)))                                         NOTI: (L2)
```

Step: 3. We load the theorem `RAT-CRITERION` from the database.

```
OMEGA: import-ass

ASS-NAME (THY-ASSUMPTION) A name of an assumption to be imported from the problem theory: rat-criterion

OMEGA: show-pds

L1            (L1)            ! (RAT (SQRT 2))                                                     HYP
               ...
L2            (L1)            ! FALSE                                                             OPEN

RAT-CRITERION (RAT-CRITERION) ! (FORALL-SORT ([X].                                                 THM
                                  (EXISTS-SORT ([Y].
                                    (EXISTS-SORT ([Z].
                                      (AND (= (TIMES X Y) Z)
                                           (NOT (EXISTS-SORT ([D]. (COMMON-DIVISOR Y Z D))
                                                             INT))))
                                      INT))
                                    INT))
                                  RAT)

SQRT2-NOT-RAT ()              ! (NOT (RAT (SQRT 2)))                                         NOTI: (L2)
```

Step: 4. We instantiate the (sorted) universal quantifier of `RAT-CRITERION` with term `(sqrt 2)`. Thereby we employ the information in `L1` saying that `(SQRT 2)` is of sort `RAT`.

```
OMEGA: foralle-sort

UNIV-LINE (NDLINE) Universal line: [RAT-CRITERION]

LINE (NDLINE) A line: [()]

TERM (TERM) Term to substitute: (sqrt 2)

SO-LINE (NDLINE) A line with sort: [L1];;;CSM Arbitrary [2]: 0 provers have to be killed

OMEGA: show-line* (rat-criterion l3)

RAT-CRITERION (RAT-CRITERION) ! (FORALL-SORT ([X].                                                 THM
                                  (EXISTS-SORT ([Y].
                                    (EXISTS-SORT ([Z].
                                      (AND (= (TIMES X Y) Z)
                                           (NOT (EXISTS-SORT ([D].(COMMON-DIVISOR Y Z D))
                                                             INT))))
                                      INT))
                                    INT))
                                  RAT)

L3         (L1)               ! (EXISTS-SORT ([DC-4248].                     FORALLE-SORT: ((SQRT 2)) (RAT-CRITERION L1)
                                  (EXISTS-SORT ([DC-4251].
                                    (AND (= (TIMES (SQRT 2) DC-4248)
                                            DC-4251)
                                         (NOT (EXISTS-SORT ([DC-4255]. (COMMON-DIVISOR DC-4248 DC-4251 DC-4255))
                                                           INT))))
                                    INT))
                                  INT)
```

Step: 5. We eliminate the first (sorted) existential quantifier by introducing constant `n`. This generates the additional information that `n` is of sort integer in line `L4`.

```
OMEGA: existse-sort

EX-LINE (NDLINE) An existential line: [L3]

LINE (NDLINE) A line to be proved: [L2]

PARAM (TERMSYM) A term: [dc-42481]n

PREM (NDLINE) The second premise line: [()]
;;;CSM Arbitrary [2]: 0 provers have to be killed


OMEGA: show-line* (l2 l3 l4 l5)


L2        (L1)            ! FALSE                                                          EXISTSE-SORT: (N) (L3 L5)


L3        (L1)            ! (EXISTS-SORT ([DC-4248].                           FORALLE-SORT: ((SQRT 2)) (RAT-CRITERION L1)
                              (EXISTS-SORT ([DC-4251].
                                (AND (= (TIMES (SQRT 2) DC-4248)
                                        DC-4251)
                                     (NOT (EXISTS-SORT ([DC-4255]. (COMMON-DIVISOR DC-4248 DC-4251 DC-4255))
                                                       INT))))
                                INT))
                              INT)


L4        (L4)            ! (AND (INT N)                                                                        HYP
                                 (EXISTS-SORT ([DC-4260].
                                   (AND (= (TIMES (SQRT 2) N)
                                           DC-4260)
                                        (NOT (EXISTS-SORT ([DC-4264]. (COMMON-DIVISOR N DC-4260 DC-4264))
                                                          INT))))
                                   INT))


L5        (L4 L1)         ! FALSE                                                                              OPEN
```

Step: 6. We split the obtained conjunction in `L4` in its conjuncts.

```
OMEGA: ande

CONJUNCTION (NDLINE) Conjunction to split: [L4]

LCONJ (NDLINE) Left conjunct: [()]

RCONJ (NDLINE) Right conjunct: [()]
;;;CSM Arbitrary [2]: 0 provers have to be killed


OMEGA: show-line* (l6 l7)


L6        (L4)            ! (INT N)                                                                      ANDE: (L4)


L7        (L4)            ! (EXISTS-SORT ([DC-4260].                                                     ANDE: (L4)
                              (AND (= (TIMES (SQRT 2) N)
                                      DC-4260)
                                   (NOT (EXISTS-SORT ([DC-4264]. (COMMON-DIVISOR N DC-4260 DC-4264))
                                                     INT))))
                              INT)
```

Step: 7. We eliminate the second (sorted) existential quantifier by introducing constant `m`. This introduces the conjunction in line `L8`.

```
OMEGA: existse-sort

EX-LINE (NDLINE) An existential line: [L3]l7

LINE (NDLINE) A line to be proved: [L5]

PARAM (TERMSYM) A term: [dc-42601]m

PREM (NDLINE) The second premise line: [()]
;;;CSM Arbitrary [2]: 0 provers have to be killed


OMEGA: show-line* (l7 l5 l8 l9)
```

```
L7    (L4)        ! (EXISTS-SORT ([DC-4260].                                      ANDE: (L4)
                     (AND (= (TIMES (SQRT 2) N)
                             DC-4260)
                          (NOT (EXISTS-SORT ([DC-4264]. (COMMON-DIVISOR N DC-4260 DC-4264))
                                            INT))))
                     INT)

L5    (L4 L1)     ! FALSE                                              EXISTSE-SORT: (M) (L7 L9)

L8    (L8)        ! (AND (INT M)                                                         HYP
                       (AND (= (TIMES (SQRT 2) N)
                               M)
                            (NOT (EXISTS-SORT ([DC-4270]. (COMMON-DIVISOR N M DC-4270))
                                              INT))))

L9    (L8 L4 L1)  ! FALSE                                                               OPEN
```

Step: 8. We split the conjunction in line `L8` in its multiple conjuncts.

```
OMEGA: ande*

CONJUNCT-LIST (NDLINE) Premises to split: l8

CONJUNCTION (NDLINE-LIST) List of conjuncts: ()
;;;CSM Arbitrary [2]: 0 provers have to be killed


OMEGA:
OMEGA: show-line* (l10 l11 l12)

L10   (L8)        ! (INT M)                                                      ANDE*: (L8)

L11   (L8)        ! (= (TIMES (SQRT 2) N) M)                                     ANDE*: (L8)

L12   (L8)     ! (NOT (EXISTS-SORT ([DC-4270]. (COMMON-DIVISOR N M DC-4270))     ANDE*: (L8)
                                   INT))
```

Step: 9. We want to infer from `(= (TIMES (SQRT 2) N) M)` in `L11` that `(= (POWER M 2) (TIMES 2 (POWER N 2)))`. For this we anticpate the later formula by introducing it as a lemma for the current open subgoal `L9`. Thereby the support nodes of `L9` become automatically available as support lines for the introduced lemma.

```
OMEGA: lemma

NODE (NDPLANLINE) An open node: [L9]

FORMULA (FORMULA) Formula to be proved as lemma: (= (power m 2) (times 2 (power n 2)))

OMEGA: show-line* (l13)

L13   (L8 L4 L1)  ! (= (POWER M 2) (TIMES 2 (POWER N 2)))                                OPEN
```

Step: 10. The lemma is proven by applying the computer algebra system Maple; the command for this is `BY-COMPUTATION`. The computation problem is passed from Ωmega to the mathematical software bus MathWeb, which in turn passes the problem to an available instance of Maple.

```
OMEGA: by-computation

LINE1 (NDLINE) A line an arithmetic term to justify.: l13

LINE2 (NDLINE-LIST) A list containing premises to be used.: (l11)

OMEGA:
OMEGA: ;;;CSM Arbitrary [2]: 0 provers have to be killed


OMEGA: show-line* (l11 l13)

L11   (L8)        ! (= (TIMES (SQRT 2) N) M)                                     ANDE*: (L8)

L13   (L8 L4 L1)  ! (= (POWER M 2) (TIMES 2 (POWER N 2)))            BY-COMPUTATION: (L11)
```

LaTeX presentation of unexpanded proof of `sqrt2-not-rat`

L19.	L19	$\vdash [\mathbb{Z}_{[\nu\to o]}(K_{[\nu]}) \wedge M_{[\nu]} = (2\cdot_{[(\nu,\nu)\to\nu]} K)]$	(Hyp)
L22.	ECD, L19	$\vdash M = (2\cdot K)$	($\wedge E$ L19)
L21.	ECD, L19	$\vdash \mathbb{Z}(K)$	($\wedge E$ L19)
L8.	L8	$\vdash [\mathbb{Z}(M) \wedge [(\sqrt{_{[\nu\to\nu]}2}\cdot N_{[\nu]}) = M \wedge \neg\exists X_{359[\nu]} : \mathbb{Z}_{[\nu\to o]}\blacksquare \text{Common-Divisor}_{[(\nu,\nu,\nu)\to o]}(N, M, X_{359})]]$	(Hyp)
L12.	$\mathcal{H}_1$	$\vdash \neg\exists X_{359[\nu]} : \mathbb{Z}\blacksquare \text{Common-Divisor}(N, M, X_{359})$	($\wedge E*$ L8)
L11.	$\mathcal{H}_1$	$\vdash (\sqrt{2}\cdot N) = M$	($\wedge E*$ L8)
L10.	$\mathcal{H}_1$	$\vdash \mathbb{Z}(M)$	($\wedge E*$ L8)
L4.	L4	$\vdash [\mathbb{Z}(N) \wedge \exists X_{2[\nu]} : \mathbb{Z}\blacksquare [(\sqrt{2}\cdot N) = X_2 \wedge \neg\exists X_{3[\nu]} : \mathbb{Z}\blacksquare \text{Common-Divisor}(N, X_2, X_3)]]$	(Hyp)
L7.	$\mathcal{H}_2$	$\vdash \exists X_{2[\nu]} : \mathbb{Z}\blacksquare [(\sqrt{2}\cdot N) = X_2 \wedge \neg\exists X_{3[\nu]} : \mathbb{Z}\blacksquare \text{Common-Divisor}(N, X_2, X_3)]$	($\wedge E$ L4)
L6.	$\mathcal{H}_2$	$\vdash \mathbb{Z}(N)$	($\wedge E$ L4)
L1.	L1	$\vdash \mathbb{Q}_{[\nu\to o]}(\sqrt{2})$	(Hyp)
L2.	$\mathcal{H}_3$	$\vdash \bot_{[o]}$	(Existse-Sort L3,L5)
RC.	RC	$\vdash \forall X_{4[\nu]} : \mathbb{Q}_{[\nu\to o]}\blacksquare \exists X_{5[\nu]} : \mathbb{Z}, X_{6[\nu]} : \mathbb{Z}\blacksquare [(X_4\cdot X_5) = X_6 \wedge \neg\exists X_{7[\nu]} : \mathbb{Z}\blacksquare \text{Common-Divisor}(X_5, X_6, X_7)]$	(Thm)
L9.	$\mathcal{H}_4$	$\vdash \bot$	(Existse-Sort L18,L20)
L5.	$\mathcal{H}_5$	$\vdash \bot$	(Existse-Sort L7,L9)
L3.	$\mathcal{H}_3$	$\vdash \exists X_{8[\nu]} : \mathbb{Z}, X_{9[\nu]} : \mathbb{Z}\blacksquare [(\sqrt{2}\cdot X_8) = X_9 \wedge \neg\exists X_{10[\nu]} : \mathbb{Z}\blacksquare \text{Common-Divisor}(X_8, X_9, X_{10})]$	(Foralle-Sort RC,L1)
L13.	$\mathcal{H}_4$	$\vdash (M\hat{}_{[(\nu,\nu)\to\nu]}2) = (2\cdot(N\hat{}2))$	(By-Computation L11)
L15.	$\mathcal{H}_4$	$\vdash \exists X_{11[\nu]} : \mathbb{Z}\blacksquare (M\hat{}2) = (2\cdot X_{11})$	(Existsi-Sort L13,L16)
L14.	$\mathcal{H}_4$	$\vdash \text{Evenp}_{[\nu\to o]}((M\hat{}2))$	(Defni L15)
L16.	$\mathcal{H}_4$	$\vdash \mathbb{Z}((N\hat{}2))$	(Wellsorted L6)
SE.	SE	$\vdash \forall X_{12[\nu]} : \mathbb{Z}\blacksquare [\text{Evenp}((X_{12}\hat{}2)) \Leftrightarrow \text{Evenp}(X_{12})]$	(Thm)
L20.	$\mathcal{H}_6$	$\vdash \bot$	(Weaken L29)
L17.	$\mathcal{H}_4$	$\vdash \text{Evenp}(M)$	(Assert SE,L10,L14)
L18.	$\mathcal{H}_4$	$\vdash \exists X_{13[\nu]} : \mathbb{Z}\blacksquare M = (2\cdot X_{13})$	(Defne L17)
L23.	$\mathcal{H}_6$	$\vdash (N\hat{}2) = (2\cdot(K\hat{}2))$	(By-Computation L13, L22)
L25.	$\mathcal{H}_6$	$\vdash \exists X_{14[\nu]} : \mathbb{Z}\blacksquare (N\hat{}2) = (2\cdot X_{14})$	(Existsi-Sort L23,L26)
L24.	$\mathcal{H}_6$	$\vdash \text{Evenp}((N\hat{}2))$	(Defni L25)
L27.	$\mathcal{H}_6$	$\vdash \text{Evenp}(N)$	(Assert SE,L6,L24)
L26.	$\mathcal{H}_6$	$\vdash \mathbb{Z}((K\hat{}2))$	(Wellsorted L21)
ECD.	ECD	$\vdash \forall X_{15[\nu]} : \mathbb{Z}, X_{16[\nu]} : \mathbb{Z}\blacksquare [[\text{Evenp}(X_{15}) \wedge \text{Evenp}(X_{16})] \Rightarrow \text{Common-Divisor}(X_{15}, X_{16}, 2)]$	(Thm)
L28.	$\mathcal{H}_6$	$\vdash \mathbb{Z}(2)$	(Wellsorted)
L29.	$\mathcal{H}_6$	$\vdash \bot$	(Assert ECD,L10,L6, L12,L17,L27,L28)
S2NR.	$\mathcal{H}_7$	$\vdash \neg\mathbb{Q}(\sqrt{2})$	($\neg I$ L2)

S2NR = Sqrt2-Not-Rat; ECD = Even-Common-Divisor; SE = Square-Even; RC = Rat-Criterion;
$\mathcal{H}_1$ = ECD, SE, L8; $\mathcal{H}_2$ = ECD, SE, L4; $\mathcal{H}_3$ = ECD, RC, SE, L1; $\mathcal{H}_4$ = ECD, RC, SE, L1, L4, L8;
$\mathcal{H}_5$ = ECD, RC, SE, L1, L4; $\mathcal{H}_6$ = ECD, RC, SE, L1, L4, L8, L19; $\mathcal{H}_7$ = ECD, RC, SE

P.rex natural language presentation of unexpanded proof of `sqrt2-not-rat`

Theorem 1. *Let 2 be a common divisor of x and y if x is even and y is even for all $y \in \mathbb{Z}$ for all $x \in \mathbb{Z}$. Let x be even if and only if x^2 is even for all $x \in \mathbb{Z}$. Let there be a $y \in \mathbb{Z}$ such that there exists a $z \in \mathbb{Z}$ such that $x \cdot y = z$ and there is no $d \in \mathbb{Z}$ such that d is a common divisor of y and z for all $x \in \mathbb{Q}$. Then $\sqrt{2}$ isn't rational.*

Proof. Let 2 be a common divisor of x and y if x is even and y is even for all $y \in \mathbb{Z}$ for all $x \in \mathbb{Z}$. Let x be even if and only if x^2 is even for all $x \in \mathbb{Z}$. Let there be a $y \in \mathbb{Z}$ such that there is a $z \in \mathbb{Z}$ such that $x \cdot y = z$ and there is no $d \in \mathbb{Z}$ such that d is a common divisor of y and z for all $x \in \mathbb{Q}$.

We prove that $\sqrt{2}$ isn't rational by a contradiction. Let $\sqrt{2}$ be rational.

Let $n \in \mathbb{Z}$ and let there be a $dc_{269} \in \mathbb{Z}$ such that $\sqrt{2} \cdot n = dc_{269}$ and there doesn't exist a $dc_{273} \in \mathbb{Z}$ such that dc_{273} is a common divisor of n and dc_{269}.

Let $m \in \mathbb{Z}$, let $\sqrt{2} \cdot n = m$ and let there be no $dc_{279} \in \mathbb{Z}$ such that dc_{279} is a common divisor of n and m.

We prove that $m^2 = 2 \cdot n^2$ in order to prove that there exists a $dc_{287} \in \mathbb{Z}$ such that $m^2 = 2 \cdot dc_{287}$. $m^2 = 2 \cdot n^2$ since $\sqrt{2} \cdot n = m$.

Therefore m^2 is even. That implies that m is even because $m \in \mathbb{Z}$. That leads to the existence of a $dc_{343} \in \mathbb{Z}$ such that $m = 2 \cdot dc_{343}$.

Let $k \in \mathbb{Z}$ and let $m = 2 \cdot k$. $2 \in \mathbb{Z}$.

We prove that $n^2 = 2 \cdot k^2$ in order to prove that there is a $dc_{353} \in \mathbb{Z}$ such that $n^2 = 2 \cdot dc_{353}$. $n^2 = 2 \cdot k^2$ since $m^2 = 2 \cdot n^2$ and $m = 2 \cdot k$.

That implies that n^2 is even. That implies that n is even since $n \in \mathbb{Z}$. Therefore we have a contradiction since $m \in \mathbb{Z}$, $n \in \mathbb{Z}$, there doesn't exist a $dc_{279} \in \mathbb{Z}$ such that dc_{279} is a common divisor of n and m, m is even and $2 \in \mathbb{Z}$.

PDS proof object of unexpanded proof of `sqrt2-not-rat`

```
(PDS (problem SQRT2-NOT-RAT)
  (in REAL)
 (declarations (type-variables )(type-constants )
    (constants  (K NUM) (N NUM) (M NUM))(meta-variables )(variables ))
   (conclusion SQRT2-NOT-RAT)
   (assumptions)
   (open-nodes)
   (support-nodes EVEN-COMMON-DIVISOR SQUARE-EVEN RAT-CRITERION)
   (nodes
     (L19 (L19) (AND (INT K) (= M (TIMES 2 K)))
      (0 ("HYP" () () "grounded" () ()))
      )
     (L22 (EVEN-COMMON-DIVISOR L19) (= M (TIMES 2 K))
      (0 ("ANDE" () (L19) "unexpanded" ()
          ("L21" "NONEXISTENT" "EXISTENT")))
      )
     (L21 (EVEN-COMMON-DIVISOR L19) (INT K)
      (0 ("ANDE" () (L19) "unexpanded" ()
          ("NONEXISTENT" "L22" "EXISTENT")))
      )
     (L8 (L8) (AND (INT M) (AND (= (TIMES (SQRT 2) N) M) (NOT (EXISTS-SORT (lam (VAR76 NUM) (COMMON-DIVISOR N M VAR76)) INT))))
      (0 ("HYP" () () "grounded" () ()))
      )
     (L12 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8) (NOT (EXISTS-SORT (lam (VAR76 NUM) (COMMON-DIVISOR N M VAR76)) INT))
      (0 ("ANDE*" () (L8) "unexpanded" ()
          ("L10" "L11" "NONEXISTENT" "EXISTENT")))
      )
     (L11 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8) (= (TIMES (SQRT 2) N) M)
      (0 ("ANDE*" () (L8) "unexpanded" ()
          ("L10" "NONEXISTENT" "L12" "EXISTENT")))
      )
     (L10 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8) (INT M)
      (0 ("ANDE*" () (L8) "unexpanded" ()
          ("NONEXISTENT" "L11" "L12" "EXISTENT")))
      )
```

```
(L4 (L4) (AND (INT N) (EXISTS-SORT (lam (VAR79 NUM) (AND (= (TIMES (SQRT 2) N) VAR79)
 (NOT (EXISTS-SORT (lam (VAR80 NUM) (COMMON-DIVISOR N VAR79 VAR80)) INT)))) INT))
 (0 ("HYP" () () "grounded" () ()))
 )
(L7 (EVEN-COMMON-DIVISOR SQUARE-EVEN L4) (EXISTS-SORT (lam (VAR79 NUM) (AND (= (TIMES
 (SQRT 2) N) VAR79) (NOT (EXISTS-SORT (lam (VAR80 NUM) (COMMON-DIVISOR N VAR79 VAR80)) INT)))) INT)
 (0 ("ANDE" () (L4) "unexpanded" ()
     ("L6" "NONEXISTENT" "EXISTENT")))
 )
(L6 (EVEN-COMMON-DIVISOR SQUARE-EVEN L4) (INT N)
 (0 ("ANDE" () (L4) "unexpanded" ()
     ("NONEXISTENT" "L7" "EXISTENT")))
 )
(L1 (L1) (RAT (SQRT 2))
 (0 ("HYP" () () "grounded" () ()))
 )
(L2 (EVEN-COMMON-DIVISOR SQUARE-EVEN RAT-CRITERION L1) FALSE
 (0 ("EXISTSE-SORT" ((:pds-term N)) (L3 L5) "unexpanded" ()
     ("EXISTENT" "EXISTENT" "NONEXISTENT")))
 )
(RAT-CRITERION (RAT-CRITERION) (FORALL-SORT (lam (VAR81 NUM) (EXISTS-SORT (lam (VAR82 NUM)
 (EXISTS-SORT (lam (VAR83 NUM) (AND (= (TIMES VAR81 VAR82) VAR83) (NOT (EXISTS-SORT
 (lam (VAR84 NUM) (COMMON-DIVISOR VAR82 VAR83 VAR84)) INT)))) INT)) INT)) RAT)
 (0 ("THM" () () "grounded" () ()))
 )
(L9 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) FALSE
 (0 ("EXISTSE-SORT" ((:pds-term K)) (L18 L20) "unexpanded" ()
     ("EXISTENT" "EXISTENT" "NONEXISTENT")))
 )
(L5 (EVEN-COMMON-DIVISOR SQUARE-EVEN L4 RAT-CRITERION L1) FALSE
 (0 ("EXISTSE-SORT" ((:pds-term M)) (L7 L9) "unexpanded" ()
     ("EXISTENT" "EXISTENT" "NONEXISTENT")))
 )
(L3 (EVEN-COMMON-DIVISOR SQUARE-EVEN RAT-CRITERION L1) (EXISTS-SORT (lam (VAR85 NUM)
 (EXISTS-SORT (lam (VAR86 NUM) (AND (= (TIMES (SQRT 2) VAR85) VAR86) (NOT (EXISTS-SORT
 (lam (VAR87 NUM) (COMMON-DIVISOR VAR85 VAR86 VAR87)) INT)))) INT)) INT)
 (0 ("FORALLE-SORT" ((:pds-term (SQRT 2))) (RAT-CRITERION L1) "unexpanded"
     () ("NONEXISTENT" "EXISTENT" "EXISTENT")))
 )
(L13 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (= (POWER M 2)
 (TIMES 2 (POWER N 2)))
 (0 ("BY-COMPUTATION" () (L11) "unexpanded" ()
     ("EXISTENT" "EXISTENT")))
 )
(L15 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EXISTS-SORT (lam (VAR88 NUM)
 (= (POWER M 2) (TIMES 2 VAR88))) INT)
 (0 ("EXISTSI-SORT" ((:pds-term (POWER N 2))((:pds-post-obj (position 2 2)))) (L13 L16)
 "unexpanded"
     () ("EXISTENT" "EXISTENT" "EXISTENT")))
 )
(L14 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EVENP (POWER M 2))
 (0 ("DefnI" ((:pds-term EVENP)(:pds-term (lam (X NUM) (EXISTS-SORT (lam (Y NUM)
 (= X (TIMES 2 Y))) INT)))(:pds-post-obj (position 0))) (L15) "grounded"
     () ("EXISTENT" "NONEXISTENT")))
 )
(L16 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (INT (POWER N 2))
 (0 ("WELLSORTED" ((((:pds-term (POWER N (S (S ZERO))))(:pds-sort INT)(:pds-symbol
  POWER-INT-CLOSED))((:pds-term (S (S ZERO)))(:pds-sort INT)(:pds-symbol NAT-INT))
  ((:pds-term (S (S ZERO)))(:pds-sort NAT)(:pds-symbol SUCC-NAT))((:pds-term (S ZERO))
  (:pds-sort NAT)(:pds-symbol SUCC-NAT))((:pds-term ZERO)(:pds-sort NAT)
  (:pds-symbol ZERO-NAT)))) (L6) "unexpanded"
     () ("EXISTENT" "EXISTENT")))
 )
(SQUARE-EVEN (SQUARE-EVEN) (FORALL-SORT (lam (VAR89 NUM) (EQUIV (EVENP (POWER VAR89 2))
 (EVENP VAR89))) INT)
 (0 ("THM" () () "grounded" () ()))
 )
(L20 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) FALSE
 (0 ("WEAKEN" () (L29) "grounded" () ("EXISTENT" "EXISTENT")))
 )
(L17 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EVENP M)
 (0 ("ASSERT" ((:pds-term (EVENP M))(:pds-nil)) (SQUARE-EVEN L10 L14) "unexpanded"
     () ("NONEXISTENT" "EXISTENT" "EXISTENT" "EXISTENT")))
 )
(L18 (EVEN-COMMON-DIVISOR SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EXISTS-SORT (lam (VAR90 NUM)
 (= M (TIMES 2 VAR90))) INT)
 (0 ("DefnE" ((:pds-term EVENP)(:pds-term (lam (X NUM) (EXISTS-SORT (lam (Y NUM)
 (= X (TIMES 2 Y))) INT)))(:pds-post-obj (position 0))) (L17) "grounded"
     () ("NONEXISTENT" "EXISTENT")))
 )
(L23 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (= (POWER N 2) (TIMES 2 (POWER K 2)))
 (0 ("BY-COMPUTATION" () (L13 L22) "unexpanded" ()
     ("EXISTENT" "EXISTENT" "EXISTENT")))
 )
(L25 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EXISTS-SORT (lam
 (VAR91 NUM) (= (POWER N 2) (TIMES 2 VAR91))) INT)
 (0 ("EXISTSI-SORT" ((:pds-term (POWER K 2))((:pds-post-obj (position 2 2)))) (L23 L26)
 "unexpanded"
     () ("EXISTENT" "EXISTENT" "EXISTENT")))
 )
```

```
  (L24 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EVENP (POWER N 2))
   (0 ("DefnI" ((:pds-term EVENP)(:pds-term (lam (X NUM) (EXISTS-SORT (lam (Y NUM)
   (= X (TIMES 2 Y))) INT)))(:pds-post-obj (position 0))) (L25) "grounded"
       () ("EXISTENT" "NONEXISTENT")))
   )
  (L27 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (EVENP N)
   (0 ("ASSERT" ((:pds-term (EVENP N))(:pds-nil)) (SQUARE-EVEN L6 L24) "unexpanded"
       () ("NONEXISTENT" "EXISTENT" "EXISTENT" "EXISTENT")))
   )
  (L26 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (INT (POWER K 2))
   (0 ("WELLSORTED" ((((:pds-term (POWER K (S (S ZERO))))(:pds-sort INT)(:pds-symbol POWER-INT-CLOSED))
   ((:pds-term (S (S ZERO)))(:pds-sort INT)(:pds-symbol NAT-INT))((:pds-term (S (S ZERO)))
   (:pds-sort NAT)(:pds-symbol SUCC-NAT))((:pds-term (S ZERO))(:pds-sort NAT)(:pds-symbol SUCC-NAT))
   ((:pds-term ZERO)(:pds-sort NAT)(:pds-symbol ZERO-NAT)))) (L21) "unexpanded"
       () ("EXISTENT" "EXISTENT")))
   )
  (EVEN-COMMON-DIVISOR (EVEN-COMMON-DIVISOR) (FORALL-SORT (lam (VAR92 NUM) (FORALL-SORT
   (lam (VAR93 NUM) (IMPLIES (AND (EVENP VAR92) (EVENP VAR93)) (COMMON-DIVISOR VAR92 VAR93 2)))
   INT)) INT)
   (0 ("THM" () () "grounded" () ()))
   )
  (L28 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) (INT 2)
   (0 ("WELLSORTED" ((((:pds-term (S (S ZERO)))(:pds-sort INT)(:pds-symbol NAT-INT))
   ((:pds-term (S (S ZERO)))(:pds-sort NAT)(:pds-symbol SUCC-NAT))((:pds-term (S ZERO))
   (:pds-sort NAT)(:pds-symbol SUCC-NAT))((:pds-term ZERO)(:pds-sort NAT)(:pds-symbol ZERO-NAT))))
   () "unexpanded"
       () ("EXISTENT")))
   )
  (L29 (EVEN-COMMON-DIVISOR L19 SQUARE-EVEN L8 L4 RAT-CRITERION L1) FALSE
   (0 ("ASSERT" ((:pds-term FALSE)(:pds-nil)) (EVEN-COMMON-DIVISOR L10 L6 L12 L17 L27 L28)
    "unexpanded"
       ()
       ("NONEXISTENT" "EXISTENT" "EXISTENT" "EXISTENT" "EXISTENT" "EXISTENT" "EXISTENT"
        "EXISTENT")))
   )
  (SQRT2-NOT-RAT (EVEN-COMMON-DIVISOR SQUARE-EVEN RAT-CRITERION) (NOT (RAT (SQRT 2)))
   (0 ("NOTI" () (L2) "grounded" () ("EXISTENT" "NONEXISTENT")))
   ))
(lemmata)
(agenda)
(controls
  (L19 (() () () ()))
  (L22 (() () () ()))
  (L21 (() () () ()))
  (L8 (() () () ()))
  (L12 (() () () ()))
  (L11 (() () () ()))
  (L10 (() () () ()))
  (L4 (() () () ()))
  (L7 (() () () ()))
  (L6 (() () () ()))
  (L1 (() () () ()))
  (L2 ((L3 L1) () () ()))
  (RAT-CRITERION (() () () ()))
  (L9 ((L16 L6 L7 L4 L1 L3 L8 L12 L11 L10 L13 L14 L17 L18) (L5 L2 L17) () ()))
  (L5 ((L6 L7 L4 L1 L3) (L2) () ()))
  (L3 (() () () ()))
  (L13 ((L10 L11 L12 L8 L3 L1 L4 L7 L6) () () ()))
  (L15 ((L13 L10 L11 L12 L8 L3 L1 L4 L7 L6) () () ()))
  (L14 ((L13 L10 L11 L12 L8 L3 L1 L4 L7 L6) () () ()))
  (L16 ((L6 L7 L4 L1 L3 L8 L12 L11 L10 L13 L14) () () ()))
  (SQUARE-EVEN (() () () ()))
  (L20 ((L29 L28 L27 L24 L23 L21 L22 L19 L18 L14 L13 L10 L11 L12 L8 L3 L1 L4 L7 L6 L16 L26 L17)
  (L9 L5 L2) () ()))
  (L17 ((L26 L16 L6 L7 L4 L1 L3 L8 L12 L11 L10 L13 L14 L17 L18 L19 L22 L21 L23 L24 L27 L28)
  (L17 L2 L5 L9) () ()))
  (L18 (() () () ()))
  (L23 ((L21 L22 L19 L18 L14 L13 L10 L11 L12 L8 L3 L1 L4 L7 L6 L16) () () ()))
  (L25 ((L23 L21 L22 L19 L18 L14 L13 L10 L11 L12 L8 L3 L1 L4 L7 L6 L16) () () ()))
  (L24 ((L23 L21 L22 L19 L18 L14 L13 L10 L11 L12 L8 L3 L1 L4 L7 L6 L16) () () ()))
  (L27 (() () () ()))
  (L26 ((L16 L6 L7 L4 L1 L3 L8 L12 L11 L10 L13 L14 L18 L19 L22 L21 L23 L24) () () ()))
  (EVEN-COMMON-DIVISOR (() () () ()))
  (L28 ((L27 L24 L23 L21 L22 L19 L18 L14 L13 L10 L11 L12 L8 L3 L1 L4 L7 L6 L16 L26) () () ()))
  (L29 (() () () ()))
  (SQRT2-NOT-RAT (() () () ())))
(plan-steps (SQRT2-NOT-RAT 0 L1 0 L2 0) (L3 0 RAT-CRITERION 0 L1 0)
  (L2 0 L4 0 L3 0 L5 0) (L6 0 L4 0) (L7 0 L4 0)
  (L5 0 L8 0 L7 0 L9 0) (L10 0 L8 0) (L11 0 L8 0) (L12 0 L8 0)
  (L13 0 L11 0) (L14 0 L15 0) (L16 0 L6 0) (L15 0 L13 0 L16 0)
  (L17 0 SQUARE-EVEN 0 L10 0 L14 0) (L18 0 L17 0)
  (L9 0 L19 0 L18 0 L20 0) (L21 0 L19 0) (L22 0 L19 0)
  (L23 0 L13 0 L22 0) (L24 0 L25 0) (L26 0 L21 0)
  (L25 0 L23 0 L26 0) (L27 0 SQUARE-EVEN 0 L6 0 L24 0) (L28 0)
  (L29 0 EVEN-COMMON-DIVISOR 0 L10 0 L6 0 L12 0 L17 0 L27 0 L28 0)
  (L20 0 L29 0) ))
```

15.4 System

What is the home page of the system?

`<http://www.ags.uni-sb.de/˜omega/>`

What are the books about the system?
There is a book forthcoming and there are several journal and conference publications. An overview of recent publications gives the homepage. The most recent overview paper on the Ωmega project is:

[1] J. Siekmann and C. Benzmüller, OMEGA: Computer Supported Mathematics. In S. Biundo, T. Frühwirth, and G. Palm (eds.), KI 2004: Advances in Artificial Intelligence: 27th Annual German Conference on AI, LNAI vol. 3228, pp. 3-28, Ulm, Germany, 2004. Springer.

The article presented here represents work done in 2002. Related and more recent publications on the $\sqrt{2}$-case study in Ωmega (in particular [3] summarizes our complete work on the $\sqrt{2}$-case study):

[2] Solving the $\sqrt{2}$-problem with interactive island theorem proving:
J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, and M. Pollet. Proof Development with Ωmega: $\sqrt{2}$ Is Irrational. In M. Baaz and A. Voronkov (eds.), *Logic for Programming, Artificial Intelligence, and Reasoning — 9th International Conference, LPAR 2002*, number 2514 in LNAI, pp. 367-387, Tbilisi, Georgia, Springer Verlag, 2002.

[3] Solving the $\sqrt{2}$-problem (and its generalization) fully automatically with proof planning:
J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, I. Normann, and M. Pollet. Proof Development with Ωmega: The Irrationality of $\sqrt{2}$. In F. Kamareddine (ed.), *Thirty Five Years of Automating Mathematics*, pp. 271-314, Kluwer Academic Publishers, 2003.

What is the logic of the system?
Ωmega employs a higher-order logic based on Church's simply typed λ-calculus (with prefix-polymorphism). Furthermore, there is a simple sort mechanism in Ωmega.

What is the implementation architecture of the system?
Ωmega consists of several distributed modules that are connected via the MathWeb mathematical software bus. Different modules are written in different programming languages (e.g. the Ωmega kernel and the proof planner are written in Lisp, the graphical user interface is written in Oz).

What does working with the system look like?
There are different modes for the proof search:

1. Interactive Proof Construction with Tactics, Methods, and Calculus Rules. This is the level of proof construction displayed in this paper. Essentially it is like any other interactive, tactic-based theorem prover and definitely *not* the level we expect real mathematics to be represented. In fact, the whole purpose of the Ωmega project is to show that there is a substantially better way to prove theorems, namely at the proof planning level.
2. Interactive Island Proof Planning. In this mode, the user presents "islands", that is, intermediate statements in the potential proof sequence. The system then closes the gaps by proof planning. See [2] for a report about an island proof of the $\sqrt{2}$ problem.
3. Automated Proof Planning. In the best of all worlds the system finds a proof fully automatically by a hierarchical expansion of a high-level proof plan into a low-level logic proof. See [3] for a fully automated solution to the $\sqrt{2}$ problem, where we also introduce and automatically prove its generalization: the irrationality of $\sqrt[i]{k}$ for any $i, k \in \mathbb{N}$.
4. Calling External Reasoners via MathWeb. This is important for the $\sqrt{2}$ problem as well. Some gaps (subproblems) are closed by a computer algebra system, simple gaps in an island proof are closed by (fast and simple) first- and higher-order theorem provers and the proof planner also uses a constraint solver as an external subsystem.
5. Combinations of the above.

What is special about the system compared to other systems? The Ωmega system is a representative system in the new paradigm of *proof planning* and combines interactive and automated proof construction in *mathematical domains*.

The main purpose of the Ωmega project is to show that computer-supported mathematics and proving can be done at a more advanced and human-oriented level of abstraction as typically found in a mathematical paper or textbook. However, it can be used also just like any other interactive tactic-based system, as this article shows.

Ωmega's inference mechanism so far has been based on a higher-order natural deduction (ND) variant of a sorted version of Church's simply typed λ-calculus. The user can interactively construct proofs directly at the calculus level or at the more abstract level of *tactics* (as shown in this article) or proof planning with *methods*. Proof construction can be supported by already proven assertions and lemmata and also by calls to external systems to simplify or solve subproblems. A recent issue in the project is to replace Ωmega's ND calculus by a more human-oriented, sound and complete base framework that better and more directly supports reasoning at the logic and assertion level.

At the core of Ωmega is the *proof plan data structure* PDS in which proofs and *proof plans* are represented at various levels of granularity and abstraction. The proof plans are classified with respect to a taxonomy of mathematical theories, which are currently being replaced by the mathematical data base MBase. The user of Ωmega, or the proof planner Multi, or the suggestion mechanism Ω-Ants modify the PDS during proof development. They can invoke external reasoning systems whose results are included in the PDS after appropriate transformation. After expansion of these high level proofs to the underlying ND calculus, the PDS can be checked by Ωmega's proof checker. User interaction is supported by the graphical user interface LΩUI and the translation into natural language by the proof explainer *P.rex*.

Several first-order ATPs are connected to Ωmega via Tramp, which is a proof transformation system that transforms resolution-style proofs into assertion level ND proofs.

What are other versions of the system?
The most recent version is Ωmega 3.6.

Who are the people behind the system?
The list of Ωmega group members and affiliated researchers includes (many RAs work in Ωmega related research projects and only a few directly on the kernel of the system):

Serge Autexier, Christoph Benzmüller, Chad Brown, Mark Buckley, Lassaad Cheikhrouhou, Dominik Dietrich, Armin Fiedler, Andreas Franke, Helmut Horacek, Mateja Jamnik (now at Cambridge University, Cambridge, UK), Manfred Kerber (now at University of Birmingham, Birmingham, UK), Michael Kohlhase (now at International University Bremen, Bremen, Germany), Henri Lesourd, Andreas Meier, Erica Melis, Martin Pollet, Marvin Schiller, Jörg Siekmann, Volker Sorge (now at University of Birmingham, Birmingham, UK), Carsten Ullrich, Quoc Bao Vo (now at RMIT University, Melbourne, Australia), Marc Wagner, Claus-Peter Wirth, Jürgen Zimmer.

What are the main user communities of the system?
The Ωmega system is employed at: Saarland University and the DFKI (AG Siekmann), the University of Birmingham (Manfred Kerber and Volker Sorge), International University Bremen (Michael Kohlhase), Cambridge University (Mateja Jamnik), University of Edinburgh (Jürgen Zimmer).

What large mathematical formalizations have been done in the system?
The Ωmega system has been used in several other case studies, which illustrate in particular the interplay of the various components, such as proof planning supported by heterogeneous external reasoning systems. Publication references to these case studies are available in [1] (see above).

A typical example for a class of problems that cannot be solved by traditional automated theorem provers is the class of ϵ–δ–proofs. This class was originally proposed as a challenge by Woody Bledsoe and it comprises theorems such as LIM+ and LIM*, where LIM+ states that the limit of the sum of two functions equals the sum of their limits and LIM* makes the corresponding statement for multiplication. The difficulty of this domain arises from the need for arithmetic computation in order to find a suitable instantiation of free (existential) variables (such as a δ depending on an ϵ). Crucial for the success of Ωmega's proof planning is the integration of suitable experts for these tasks: the arithmetic computation is done by the computer algebra system Maple, and an appropriate instantiation for δ is computed by the constraint solver Cosie. We have been able to solve all challenge problems suggested by Bledsoe and many more theorems in this class taken from a standard textbook on real analysis.

Another class of problems we tackled with proof planning is concerned with residue classes. In this domain we show theorems such as: "the residue class structure $(Int_5, \bar{+})$ is associative", "it has a unit element", and similar properties, where Int_5 is the set of all congruence classes modulo 5 (i.e. $\{\bar{0}_5, \bar{1}_5, \bar{2}_5, \bar{3}_5, \bar{4}_5\}$) and $\bar{+}$ is the addition on residue classes. We have also investigated whether two given structures are isomorphic or not and altogether we have proven more than 10,000 theorems of this kind. Although the problems in the residue class domain are still within the range of difficulty a traditional automated theorem prover could handle, it was nevertheless an interesting case study for proof planning, since multi-strategy proof planning generated substantially different proofs based on entirely different proof ideas.

Another important proof technique is Cantor's diagonalization technique and we also developed methods and strategies for this class. Important theorems we have been able to prove are the undecidability of the halting problem and Cantor's theorem (cardinality of the set of subsets), the non-countability of the reals in the interval $[0, 1]$ and of the set of total functions, and similar theorems.

Finally, a good candidate for a standard proof technique are completeness proofs for refinements of resolution, where the theorem is usually first shown at the ground level using the excess-literal-number technique and then ground completeness is lifted to the predicate calculus. We have done this for many refinements of resolution with Ωmega.

What representation of the formalization has been put in this paper?
The problem has been formalized in POST syntax. The formalization employed knowledge provided in Ωmega's hierarchically structured knowledge base. However, because of the prerequisites posted for this volume, only the tactic-level proof is shown. This (logic) level of representation is important not least of all for the final proof checker. However, cognitively and practically far more important is the proof plan level of abstraction for a human-oriented mode of representation (see [2] and [3] above).

What needs to be explained about this specific proof?
Our aim was to follow our own (logic-level) proof sketch on a blackboard as closely as possible with the system. We replayed this proof idea in the system by partly employing interactive theorem proving in an island style, i.e., we anticipated some islands (some intermediate proof goals) and closed the gaps with the help of tactics and external reasoning systems. The results of the external system applications, such as the Otter proofs, have been translated and integrated into the central Ωmega proof object. This proof object can be verified

by an independent proof checker after expansion to base calculus level (natural deduction). The only tactic that can not be fully expanded and checked yet is `by-computation`, i.e. the computations contributed by the computer algebra system Maple. The translation of computer algebra proofs into natural deduction is an interesting problem on its own and we have work in progress.