De kunst van het bewijzen

Freek Wiedijk

Radboud Universiteit Nijmegen

1 Lagere en hogere wiskunde

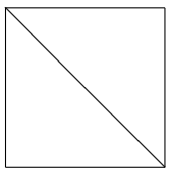
In de wiskunde kunnen we twee soorten van activiteiten onderscheiden:

- **Berekenen.** Hierbij gaat het om het *wat*.
- **Bewijzen.** Hierbij gaat het om het *waarom*.

Het gaat in deze classificatie om *groepen* van activiteiten. Zo vallen zowel het uitwerken van een merkwaardig product als het maken van een grafiek van een functie in de 'berekenen' categorie, terwijl het definiëren van een nieuw begrip in termen van al bestaande begrippen in de categorie 'bewijzen' valt. Merk op dat we met 'berekenen' niet alleen rekenen met getallen bedoelen, maar ook het symbolisch uitwerken van een antwoord (ook daarbij gaat het om het 'wat'.)

Men kan berekenen als 'lagere' wiskunde, en bewijzen als 'hogere' wiskunde beschouwen. Iemand die alleen de berekeningskant van de wiskunde kent heeft geen goed beeld van wat wiskunde is. Iemand met aanleg voor wiskunde die daarom wiskunde erg de moeite waard zou kunnen vinden, maar die de bewijzenkant van wiskunde niet kent, zal waarschijnlijk menen dat wiskunde een stuk saaier is dan het in werkelijkheid is.

Een voorbeeld van het onderscheid tussen deze twee soorten activiteit is de volgende. Bekijk het volgende plaatje van een diagonaal in een vierkant:



Hierover kunnen we de volgende twee vragen stellen:

- **Bereken** de verhouding tussen de lengte van de diagonaal van het vierkant en de lengte van de zijde van het vierkant. (Met de stelling van Pythagoras is het antwoord op deze vraag natuurlijk dat deze verhouding $\sqrt{2}$ is. Hierbij wordt de stelling van Pythagoras gebruikt als een *rekenregel*.)
- **Bewijs** dat de verhouding tussen de lengte van de diagonaal van het vierkant en de lengte van de zijde van het vierkant niet te schrijven is als de verhouding tussen twee gehele getallen. (En een bewijs hiervan is natuurlijk het klassieke bewijs van deze stelling uit de school van Pythagoras. We komen in Sectie 3 op dit bewijs terug.)

Het gaat hierbij duidelijk om twee heel verschillende soorten vragen.

Begin jaren zeventig werd er een nieuwe vorm van bewijzen ontwikkeld die *formalisatie* heet. Hierbij worden de bewijzen in de computer gecodeerd in een vorm waarbij een computerprogramma kan nagaan of deze bewijzen volledig correct zijn. Zo'n computerprogramma heet een *bewijschecker* of *bewijsassistent*.

Voordat de formalisatietechnologie was ontwikkeld was een bewijs altijd iets dat zich in een mensenhoofd afspeelde, en met schoolbord of papier aan andere wiskundigen werd doorgegeven. De wiskundigen realiseerden zich wel dat het mogelijk was bewijzen in een *formeel systeem* te representeren (dit is het onderwerp van de *mathematische logica*), maar hoewel wel geprobeerd werd dit echt te doen (het verst in de beroemde *Principia Mathematica* van Alfred North Whitehead en Bertrand Russell, dat gepubliceerd werd in 1910–1913), was het zonder computers te onpraktisch om op deze manier serieuze wiskunde weer te geven.

Begin jaren zeventig veranderde dit dus door de opkomst van de computer, en sindsdien bestaat er met formalisatie een vorm van wiskundig bewijzen die een objectieve status heeft, buiten de menselijke geest.

Met formalisatie heeft de kunst van het wiskundige bewijs een essentieel nieuwe vorm bereikt. We zullen in de rest van deze tekst een introductie geven tot formalisatie van wiskunde.

2 Wiskunde in de computer

Computerwiskunde is het gebruik van computers in de wiskunde. Dit bestaat in verschillende soorten, die significant van elkaar verschillen. Het gaat ons bij formalisatie om de vierde en laatste soort.

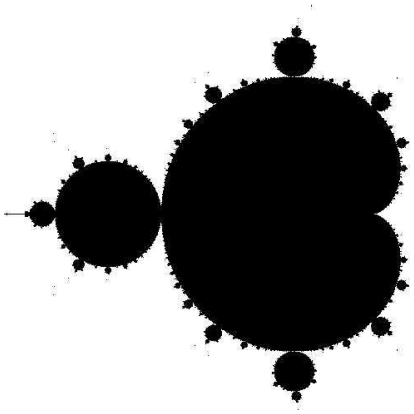
De soorten computerwiskunde zijn:

Computerwiskunde soort 1: numerieke wiskunde

Hierbij worden computers gebruikt om met *getallen* te rekenen, het zogenaamde 'number crunching'. Hieronder valt ook het gebruik van computers voor visualisatie.

Bij het rekenen met computers wordt vaak gebruik gemaakt van 'floating point' ('drijvende komma') berekeningen volgens de zogenaamde IEEE 754 standaard, waarbij na iedere rekenkundige bewerking het resultaat wordt afgerond om de uitkomsten in een eindig aantal cijfers na de komma te kunnen blijven representeren. Het zal duidelijk zijn dat hierdoor alleen een benadering van het wiskundig correcte antwoord wordt bereikt, en niet de exacte waarde.

Voor experimentatie en visualisatie is dit niet heel erg. Zo kan bijvoorbeeld een plaatje van de bekende Mandelbrot-verzameling:



(de verzameling van punten c in het complexe vlak waarvoor de rij gegeven door $x_0 = 0$ en $x_{n+1} = {x_n}^2 + c$ niet naar het oneindige wegloopt) prima worden gemaakt met floating point-berekeningen. Het feit dat bij de rand misschien door afrondfouten een enkele pixel verkeerd wordt gekleurd geeft daarbij niets.

Evenwel kun je numerieke methoden ook gebruiken om wiskundig harde informatie te verzamelen. Om dit te bereiken moeten afrondfouten expliciet worden bijgehouden, zodat er bekend is wat de relatie tussen de uitkomst van de berekening en het wiskundig correcte antwoord is.

Dit betekent dat numerieke methoden ondanks de afrondfouten wel degelijk bruikbaar zijn om harde wiskundige bewijzen mee te ondersteunen.

Een voorbeeld van een bewijs waarbij numeriek gebruik van de computer essentieel was, was de ontkrachting in 1985 door Andrew Odlyzko en Herman te Riele van het *Mertens-vermoeden*. Het Mertens-vermoeden zegt dat voor de 'Mertens-functie'

$$M(n) = \sum_{k=1}^{n} \mu(k)$$

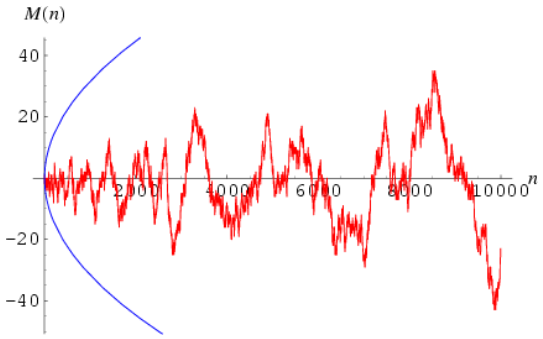
waarbij $\mu(k)$ de Möbius-functie is gedefinieerd door

$$\mu(k) = \begin{cases} 1 & \text{als } k \text{ kwadraatvrij is met een even aantal priemfactoren} \\ -1 & \text{als } k \text{ kwadraatvrij is met een oneven aantal priemfactoren} \\ 0 & \text{als } k \text{ deelbaar is door een kwadraat} \end{cases}$$

geldt dat

$$|M(n)| \leq \sqrt{n}$$

Numerieke evidentie lijkt erop te wijzen dat dit vermoeden waar is:



(het vermoeden zegt dat deze bibberlijn, die de grafiek van de Mertens-functie is, binnen de parabool blijft), maar uiteindelijk werd dus bewezen dat dit niet het geval is. Hiervoor was een ingewikkeld bewijs nodig. (Het is tot nog toe *niet* mogelijk gebleken om $M(n)$ expliciet uit te rekenen voor een n die groot genoeg is dat $|M(n)| \leq \sqrt{n}$ niet langer geldt. In plaats daarvan werd het Mertens-vermoeden via een indirecte redenering weersproken.)

Voor dit bewijs waren 2000 nulpunten in het complexe vlak van de Riemann zeta-functie

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

nodig, met een precisie van minstens 100 decimalen:

$0.5 + i \cdot 14.134725141734693790457251983562470270784257115699243175685567460149963429809256764949010393171561012 7\ldots$
$0.5 + i \cdot 21.022039638771554992628479593896902777334340524902781754629520403587598586068890799713658514180151419 5\ldots$
$0.5 + i \cdot 25.010857580145688763213790992562821818659549672557996672496542006745092098441644277840238224558062440 7\ldots$
$0.5 + i \cdot 30.424876125859513210311897530584091320181560023715440180962146036993329389333277920290584293902089110 6\ldots$
$0.5 + i \cdot 32.935061587739189690662368964074903488812715603517039009280003440784815608630551005938848496135348724 5\ldots$
$0.5 + i \cdot 37.586178158825671257217763480705332821405597350830793218333001113622149089618537264730329104945823803 4\ldots$
$0.5 + i \cdot 40.918719012147495187398126914633254395726165962777279536161303667253280528720071282996003719889546875 5\ldots$
$0.5 + i \cdot 43.327073280914999519496122165406805782645668371836871446878893685521088322305053626456349371063190933 5\ldots$
$0.5 + i \cdot 48.005150881167159727942472749427516041686844001144425117775312519814090216416308281330335372305400997 7\ldots$
$0.5 + i \cdot 49.773832477672302181916784678563724057723178299676662100781955750433511611515739278732707507400931330 0\ldots$
$\ldots$ *nog 1990 nulpunten* $\ldots$

Dat is een veel grotere precisie dan de floating point-hardware van een computer levert, maar is prima met een computer te berekenen. En de ontkrachting van het Mertens-vermoeden is dus niet alleen 'correct op afrondfouten na', maar wiskundig volkomen zeker.

Computerwiskunde soort 2: computer algebra

Bij computer algebra gaat het niet om het berekenen van getallen maar om *symbolisch* rekenen. Evenwel gaat het hier nog steeds om activiteiten uit de 'berekenen' categorie en niet uit de 'bewijzen' categorie. De bekendste computerprogramma's voor computer algebra zijn Mathematica en Maple.

Een voorbeeld van een 'berekening' die een computer algebra systeem voor je kan doen is het symbolisch uitrekenen van een integraal (dit voorbeeld is een Maple sessie):

```
> Int(ln(x)/(1 - x), x = 0..1);
```

$$\int_0^1 \frac{\ln x}{1-x}\,dx$$

```
> value(%);
```

$$-\frac{\pi^2}{6}$$

```
> evalf(%);
```

$$-1.644934068$$

Het is duidelijk dat Maple de integraal zowel symbolisch als numeriek kan uitrekenen.

Evenwel is er *niet* duidelijk *hoe* Maple aan zijn antwoord is gekomen. Het gaat hier om een berekening en niet om een bewijs.

Ook was Maple iets te onvoorzichtig met de afronding. De echte waarde van de integraal is $-1.644934066848...$ wat afrondt naar -1.644934067 en niet naar -1.644934068. Hoewel dat niet *heel* erg incorrect is, is incorrectheid van resultaten van computer algebra systemen (om verschillende redenen, maar vooral door te enthousiaste vereenvoudigingen) een algemeen fenomeen. Resultaten van computer algebra moeten dus altijd met een kritisch oog gebruikt worden.

Computer algebra systemen bevatten allerlei hele krachtige algoritmen om symbolisch te rekenen. Een voorbeeld hiervan is het *algoritme van Risch*, die *als* de uitkomst van een onbepaalde integraal in 'gesloten vorm' (een expressie in termen van de rekenkundige bewerkingen en de elementaire transcendente functies, te weten: exponentiatie, logaritme, de trigonometrische en de inverse trigonometrische functies) kan worden geschreven dit altijd zal doen, en dat ook altijd zal vertellen *of* de integraal in zo'n gesloten vorm te schrijven is. Evenwel is het algoritme van Risch zo verschrikkelijk ingewikkeld dat bestaande software tot nog toe altijd alleen maar een deel ervan heeft geïmplementeerd.

Computerwiskunde soort 3: automatische stellingenbewijzers

Bij computer algebra *berekent* de computer, maar bij automatische stellingenbewijzers levert de computer automatisch *bewijzen*. De gebruiker geeft het programma een uitspraak, en het systeem probeert hier automatisch een bewijs voor te vinden.

Er is een hele industrie van dit soort programma's, die allemaal proberen om zoveel mogelijk uitspraken automatisch te kunnen bewijzen. De bekendste programma's van dit type zijn Otter (met als recente opvolger een stellingenbewijzer met de naam 'Prover9'), Vampire en de E prover.

Om deze programma's te ontwikkelen en te vergelijken is er een database van 'problemen' gemaakt met de naam TPTP ('Thousands of Problems for Theorem Provers') waar momenteel 7.068 problemen in zitten. Ook is er jaarlijks de CASC

competitie ('CADE Systems Competition') als onderdeel van de CADE conferentie ('Conference on Automated Deduction'). Deze competitie wordt vrijwel altijd door Vampire gewonnen.

Helaas staan dit soort programma's nog in de kinderschoenen. Er zijn eigenlijk geen interessante wiskundige bewijzen die door dit soort programma's automatisch gevonden kunnen worden. Zelfs voor een kleine stapje in een bewijs, iets wat een mens onmiddellijk ziet, zijn ze vaak nog te zwak. De stellingen die dit soort programma's kunnen bewijzen zijn altijd of van het type 'puzzel', of van een vorm dat het bewijs bestaat uit het nagaan van een groot aantal gevallen, zonder dat er enige creativiteit voor nodig is.

Het gaat bij deze programma's dus eerder om onderzoek in het onderzoeksgebied van de *kunstmatige intelligentie*, dan om software die bruikbaar is om echte wiskunde mee te doen.

Er zijn *enkele* bewijzen die met de hulp van automatische stellingenbewijzers zijn gevonden. Het bekendste hiervan is het bewijs van het Robbins-vermoeden. Robbins vroeg zich af welke structuren er allemaal zijn waarin de volgende drie vergelijkingen gelden:

$$\begin{gathered} a \vee b = b \vee a \\ a \vee (b \vee c) = (a \vee b) \vee c \\ \neg(\neg(a \vee b) \vee \neg(a \vee \neg b) = a \end{gathered}$$

Een verzameling met daarop een unaire operatie $\neg$ en een binaire operatie $\vee$ die hieraan voldoet (voor alle a, b en c uit de verzameling) heet een *Robbins algebra*. Robbins vroeg zich af of iedere Robbins algebra altijd de structuur van een *Boolse algebra* heeft. Een Boolse algebra is een structuur waarin je de objecten als verzamelingen kunt representeren waarbij dan $\neg$ complementatie ten opzichte van een grootste verzameling en $\vee$ vereniging van verzamelingen zijn. Het blijkt inderdaad zo te zijn dat een Robbins algebra altijd een Boolse algebra is.

Het Robbins-vermoeden was onbewezen van 1933 tot 1996, en beroemde wiskundigen hebben er aan gewerkt zonder het op te lossen. Uiteindelijk werd het opgelost met de hulp van de automatische stellingenbewijzer EQP, een variant van Otter, die daar (in 1996) acht dagen computertijd voor nodig had. Het uiteindelijke bewijs was overigens vrij kort en bestond uit slechts 34 stappen, die zelfs door een mens gecontroleerd kunnen worden.

Hoewel bij het Robbins-vermoeden de computer (met menselijke sturing) een onopgelost probleem wist op te lossen, ging het hierbij om een probleem van het type 'bekijk zeer veel gevallen'. Zoals al gezegd zijn automatische stellingenbewijzers nauwelijks bruikbaar voor het doen van wiskunde, en alleen nuttig bij dit specifieke soort problemen.

Computerwiskunde soort 4: bewijsassistenten

Dit zijn de systemen voor *formalisatie*. Hierbij gaat het om het uitwerken van wiskundige bewijzen met de computer. Evenwel is het nu niet de computer die

het bewijs produceert, maar de mens. De computer fungeert slechts als 'domme boekhouder' die het bewijs netjes op een rijtje zet, en ervoor zorgt dat de mens zich niet vergist. Hoogstens helpt de computer soms een beetje met de eenvoudigste stapjes van het bewijs.

De meeste bewijsassistenten zijn ontwikkeld voor het doen van bewijzen in de informatica. Het gaat daarbij om het bewijzen van eigenschappen van hardware of van software. Zo worden bewijsassistenten gebruikt om te bewijzen dat microprocessors en *compilers* voor programmeertalen geen 'bugs' bevatten.

De bekendste bewijsassistenten van dit type zijn HOL, Isabelle, Coq, PVS en ACL2. Hoewel deze voor informatica-toepassingen zijn ontwikkeld, zijn ze ook bruikbaar om wiskundige bewijzen op correctheid te controleren. In de volgende sectie zullen we in detail kijken naar die systemen die specifiek voor wiskunde erg geschikt zijn.

Er kunnen dus vier soorten computerwiskunde worden onderscheiden. Idealiter zou je een systeem willen hebben waarin al deze soorten tot een geheel zijn samengevoegd. Helaas bestaat zo'n systeem nog niet. Het ontwikkelen van een dergelijk systeem is het onderwerp van een internationaal project met de naam *Calculemus*. Dit is een citaat van de laat-zeventiende eeuwse filosoof Gottfried Leibniz. Hij was één van de eersten die inzag dat redeneren in een formeel systeem mogelijk is. Hij had een wereld voor ogen waar bij inhoudelijke meningsverschillen (ook in de politiek) er zou kunnen worden *uitgerekend* wie er gelijk heeft. In dat geval zou men zeggen 'Calculemus!' ('laten we het uitrekenen!'), en zou er daarna geen onenigheid meer kunnen zijn.

Uiteraard is dit, ondanks de stormachtige ontwikkeling van de computerwiskunde, nog altijd een utopie.

3 Vier bewijsassistenten voor wiskunde

Stel je hebt een wiskundig bewijs, en je wil zeker weten dat bewijs absoluut correct is. Wat je dan moet doen is het bewijs invoeren in een bewijsassistent, die dan de correctheid van het bewijs voor je controleert. De eerste stap die je dan moet nemen is het selecteren van de bewijsassistent die je daarvoor wilt gebruiken.

Nu bestaan er veel bewijsassistenten, en zelfs al veel bewijsassistenten die voor specifiek wiskundige bewijzen geschikt zijn. Een overzicht van een aantal van deze bewijsassistenten is te vinden in het boekje *The Seventeen Provers of the World*, dat in de Springer LNAI reeks is uitgegeven als deeltje 3600. De voorbeelden in deze sectie komen uit dit boekje. De bijbehorende formalisaties kunnen worden gevonden op de webpagina:

`http://www.cs.ru.nl/~freek/comparison/`

In *The Seventeen Provers of the World* wordt voor elke bewijsassistent een bewijs van de irrationaliteit van $\sqrt{2}$ (de eigenschap van de diagonaal van het vierkant op pagina 1) als formalisatie gepresenteerd.

Er zijn dus minstens zeventien bewijsassistenten die bruikbaar zijn voor wiskunde. Hieronder zijn zowel de populaire bewijsassistenten van vandaag die boven al werden genoemd – HOL, Isabelle, Coq, PVS en ACL2 – als bewijsassistenten die specifiek zijn ontworpen voor het formaliseren van wiskunde – zoals Mizar en Metamath.

Enige tijd geleden vond ik een top 100 van 'leuke stellingen' op het Internet. In deze lijst stonden een groot aantal stellingen waarvan ik wist dat er al een bewijs van was geformaliseerd. En de irrationaliteit van $\sqrt{2}$ was zelfs de eerste in de lijst! Vandaar dat ik een onderzoekje begon naar welke van deze stellingen in welke systemen waren geformaliseerd. Het resultaat hiervan staat op de webpagina:

`http://www.cs.ru.nl/~freek/100/`

Momenteel zijn er 80 van de 100 stellingen geformaliseerd. En hier is dit aantal uitgesplitst naar bewijsassistent:

HOL Light	69
Mizar	44
ProofPower	42
Isabelle	40
Coq	39
PVS	15
ACL2	12

Het moge duidelijk zijn dat er vijf systemen zijn die echt gebruikt zijn voor het formaliseren van wiskunde: HOL Light, Mizar, ProofPower, Isabelle en Coq. Nu zijn HOL Light en ProofPower vrijwel hetzelfde systeem (ze zijn allebei herimplementaties van het HOL systeem), en heeft HOL Light een betere bibliotheek van geformaliseerde wiskunde. Daarom zullen we hier ProofPower buiten beschouwing laten.

Dat laat vier systemen over die we nu één voor één onder de loep zullen nemen: HOL Light, Isabelle, Coq en Mizar.

HOL Light

HOL is misschien wel de meest bekende bewijsassistent. Het systeem werd ontwikkeld in Engeland aan de Universiteit van Cambridge door Michael Gordon. Omdat het zo bekend is zijn er in de loop van de tijd een aantal herimplementaties van gemaakt, waaronder HOL Light en ProofPower. (Het originele HOL systeem heet nu HOL4.) Het Isabelle systeem is ook een variant van HOL, maar één die inmiddels wat meer van het originele systeem is gaan verschillen.

HOL Light werd ontwikkeld door John Harrison als onderdeel van zijn promotieonderzoek aan de Universiteit van Cambridge. Het is een zeer elegante en gestroomlijnde versie van HOL, en is door John Harrison gebruikt voor het formaliseren van allerlei stukken van de wiskunde (wat hij doet naast zijn baan bij Intel waar hij floating point processors correct bewijst).

John Harrison

Hier is hoe een bewijs er in HOL Light uitziet (dit lijkt sterk op hoe het er in de andere varianten van HOL uitziet):

```
let NSQRT_2 = prove
 (`!p q. p * p = 2 * q * q ==> q = 0`,
  MATCH_MP_TAC num_WF THEN REWRITE_TAC[RIGHT_IMP_FORALL_THM] THEN
  REPEAT STRIP_TAC THEN FIRST_ASSUM(MP_TAC o AP_TERM `EVEN`) THEN
  REWRITE_TAC[EVEN_MULT; ARITH] THEN REWRITE_TAC[EVEN_EXISTS] THEN
  DISCH_THEN(X_CHOOSE_THEN `m:num` SUBST_ALL_TAC) THEN
  FIRST_X_ASSUM(MP_TAC o SPECL [`q:num`; `m:num`]) THEN
  POP_ASSUM MP_TAC THEN CONV_TAC SOS_RULE);;

let SQRT_2_IRRATIONAL = prove
 (`~rational(sqrt(&2))`,
  SIMP_TAC[rational; real_abs; SQRT_POS_LE; REAL_POS; NOT_EXISTS_THM] THEN
  REPEAT GEN_TAC THEN DISCH_THEN(CONJUNCTS_THEN2 ASSUME_TAC MP_TAC) THEN
  DISCH_THEN(MP_TAC o AP_TERM `\x. x pow 2`) THEN
  ASM_SIMP_TAC[SQRT_POW_2; REAL_POS; REAL_POW_DIV; REAL_POW_2; REAL_LT_SQUARE;
               REAL_OF_NUM_EQ; REAL_EQ_RDIV_EQ] THEN
  ASM_MESON_TAC[NSQRT_2; REAL_OF_NUM_EQ; REAL_OF_NUM_MUL]);;
```

Het eerste lemma zegt dat voor natuurlijke getallen

$$p^2 = 2q^2$$

alleen kan gelden als

$$q = 0$$

Het zal duidelijk zijn dat de bewijzen die hier worden gecodeerd alleen te begrijpen zijn door de 'scripts' die hier staan op een computer te executeren. (We zullen hier niet uitleggen welk bewijs hier precies staat. Het voorbeeld dient slechts om een indruk te geven van hoe geformaliseerde wiskunde er uit ziet.)

De webpagina van het HOL Light systeem is:

`http://www.cl.cam.ac.uk/~jrh13/hol-light/`

Het HOL Light systeem is moeilijk te leren, en waarschijnlijk buiten het bereik van iemand zonder achtergrond in de logica, en zonder hulp van iemand die direct naast hem of haar achter de computer zit.

Isabelle

Het Isabelle systeem is ook een variant van HOL, maar is daarna verder ontwikkeld en lijkt tegenwoordig niet meer zo op de oorspronkelijke HOL. Het is ontwikkeld in Engeland door Larry Paulson van de Universiteit van Cambridge en in Duitsland door Tobias Nipkow en Markus Wenzel van de Technische Universiteit München.

Isabelle is waarschijnlijk het breedste systeem voor het formaliseren van wiskunde. Het combineert eigenschappen van allerlei andere systemen in een gebalanceerd geheel. Hier is hoe een bewijs er in Isabelle uitziet:

```
theorem sqrt_prime_irrational: "p ∈ prime ⟹ sqrt (real p) ∉ ℚ"
proof
  assume p_prime: "p ∈ prime"
  then have p: "1 < p" by (simp add: prime_def)
  assume "sqrt (real p) ∈ ℚ"
  then obtain m n where
      n: "n ≠ 0" and sqrt_rat: "|sqrt (real p)| = real m / real n"
    and gcd: "gcd (m, n) = 1" ..
  have eq: "m^2 = p * n^2"
  proof -
    from n and sqrt_rat have "real m = |sqrt (real p)| * real n" by simp
    then have "real (m^2) = (sqrt (real p))^2 * real (n^2)"
      by (auto simp add: power_two real_power_two)
    also have "(sqrt (real p))^2 = real p" by simp
    also have "... * real (n^2) = real (p * n^2)" by simp
    finally show ?thesis ..
  qed
  have "p dvd m ∧ p dvd n"
  proof
    from eq have "p dvd m^2" ..
    with p_prime show "p dvd m" by (rule prime_dvd_power_two)
    then obtain k where "m = p * k" ..
    with eq have "p * n^2 = p^2 * k^2" by (auto simp add: power_two mult_ac)
    with p have "n^2 = p * k^2" by (simp add: power_two)
    then have "p dvd n^2" ..
    with p_prime show "p dvd n" by (rule prime_dvd_power_two)
  qed
  then have "p dvd gcd (m, n)" ..
  with gcd have "p dvd 1" by simp
  then have "p ≤ 1" by (simp add: dvd_imp_le)
  with p show False by simp
qed

corollary "sqrt (real (2::nat)) ∉ ℚ"
  by (rule sqrt_prime_irrational) (rule two_is_prime)
```

Het is hopelijk duidelijk dat een Isabelle bewijs in tegenstelling tot een HOL bewijs ook zonder computer te begrijpen valt.

De webpagina van het Isabelle systeem is:

`http://isabelle.in.tum.de/`

Isabelle is makkelijker te leren dan de traditionele HOL systemen, maar er is waarschijnlijk toch nog steeds een gedegen kennis van de mathematische logica voor nodig om enigszins met Isabelle uit de voeten te kunnen.

Coq

Het Coq systeem is ontwikkeld in Frankrijk door INRIA, het Franse onderzoeksinstituut voor informatica. Oorspronkelijk is de ontwikkeling ervan begonnen door Gérard Huet en Thierry Coquand, maar in de loop van de tijd is er door vele mensen aan gewerkt.

Coq lijkt aan de ene kant sterk op HOL, in de zin dat de manier van bewijzen in beide systemen erg verwant is, en dat daardoor Coq bewijzen net als HOL bewijzen ook niet zonder computer te begrijpen zijn:

```
Theorem main_thm: forall (n p : nat), n * n = double (p * p) ->  p = 0.
intros n; pattern n; apply lt_wf_ind; clear n.
intros n H p H0.
case (eq_nat_dec n 0); intros H1.
generalize H0; rewrite H1; case p; auto; intros; discriminate.
assert (H2: even n).
apply even_is_even_times_even.
apply double_even; rewrite H0; rewrite double_div2; auto.
assert (H3: even p).
apply even_is_even_times_even.
rewrite <- (double_inv (double (div2 n * div2 n)) (p * p)).
apply double_even; rewrite double_div2; auto.
rewrite main_thm_aux; auto.
assert (H4: div2 p = 0).
apply (H (div2 n)).
apply lt_div2; apply neq_O_lt; auto.
apply double_inv; apply double_inv; (repeat rewrite main_thm_aux); auto.
rewrite (even_double p); auto; rewrite H4; auto.
Qed.
```

Evenwel zijn er ook een aantal verschillen. In twee opzichten staat Coq veel meer in de Nederlandse traditie dan de HOL varianten.

Ten eerste representeert Coq bewijzen intern met behulp van de zogenaamde *lambda-calculus*. Dit is een aanpak die zijn oorsprong heeft in één van de allereerste bewijsassistenten ooit (er werd al aan begonnen in 1968), het Automath systeem. Dit was een Nederlandse product, ontwikkeld door N.G. de Bruijn en zijn onderzoeksgroep aan de Technische Universiteit Eindhoven.

N.G. de Bruijn

Eén van de vooraanstaande experts op het gebied van de lambda-calculus is Henk Barendregt van de Radboud Universiteit Nijmegen, vandaar dat aan deze universiteit veel met Coq geformaliseerd wordt.

Ten tweede werkt Coq met een variant van wiskunde die *intuïtionisme* wordt genoemd. Deze soort wiskunde werd ontwikkeld door L.E.J. Brouwer in het begin van de twintigste eeuw om de paradoxen in de logica die toen werden ontdekt te vermijden. Evenwel is het intuïtionisme nooit meer dan een marginale stroming geworden, en is vrijwel alle hedendaagse wiskunde niet-intuïtionistisch. Om een voorbeeld te geven van hoe intuïtionisme afwijkt van gewone wiskunde: in het intuïtionisme kun je niet bewijzen dat ieder reëel getal altijd óf gelijk is aan nul, óf verschillend van nul. Een wat geavanceerdere stelling die hierdoor in het intuïtionisme onbewijsbaar is, is de *tussenwaardenstelling*: dat een continue functie van $\mathbb{R}$ naar $\mathbb{R}$ die ergens negatief en ergens anders positief is altijd een nulpunt heeft.

L.E.J. Brouwer

De intuïtionistische basis van Coq is geen reden om Coq niet te gebruiken. Dit komt omdat je in Coq ook op de gewone, niet-intuïtionistische manier kunt redeneren. Veel Coq-gebruikers, inclusief de meeste Coq-gebruikers aan de Radboud Universiteit, werken desondanks intuïtionistisch.

De webpagina van het Coq systeem is:

```
http://coq.inria.fr/
```

Coq is ongeveer even moeilijk te leren als Isabelle. Dat betekent dat je een stevige basis in de mathematische logica moet hebben voordat je met Coq aan de slag kunt gaan.

Mizar

Het Poolse systeem Mizar, ontwikkeld door Andrzej Trybulec aan de universiteit van Białystok, is een beetje een vreemde eend in de bijt tussen deze vier systemen. Ten eerste is het specifiek voor wiskunde ontworpen, terwijl de andere drie systemen in eerste instantie voor informatica-toepassingen zijn ontwikkeld. Ten tweede is het systeem lange tijd 'achter het ijzeren gordijn' ontwikkeld, zonder veel contact met de rest van de onderzoeksgemeenschap voor bewijsassistenten. Hierdoor verschilt het in veel opzichten van de andere systemen.

Andrzej Trybulec

Evenwel worden er tegenwoordig meer en meer concepten van het Mizar systeem door andere systemen overgenomen. Zo is de bewijstaal van Isabelle direct gebaseerd op de bewijstaal van Mizar:

```
theorem
  sqrt 2 is irrational
proof
```

```
 assume sqrt 2 is rational;
 then consider i being Integer, n being Nat such that
W1: n<>0 and
W2: sqrt 2=i/n and
W3: for i1 being Integer, n1 being Nat st n1<>0 & sqrt 2=i1/n1 holds n<=n1
      by RAT_1:25;
A5: i=sqrt 2*n by W1,XCMPLX_1:88,W2;
C: sqrt 2>=0 & n>0 by W1,NAT_1:19,SQUARE_1:93;
 then i>=0 by A5,REAL_2:121;
 then reconsider m = i as Nat by INT_1:16;
A6: m*m = n*n*(sqrt 2*sqrt 2) by A5
  .= n*n*(sqrt 2)^2 by SQUARE_1:def 3
  .= 2*(n*n) by SQUARE_1:def 4;
 then 2 divides m*m by NAT_1:def 3;
 then 2 divides m by INT_2:44,NEWTON:98;
 then consider m1 being Nat such that
W4: m=2*m1 by NAT_1:def 3;
 m1*m1*2*2 = m1*(m1*2)*2
    .= 2*(n*n) by W4,A6,XCMPLX_1:4;
 then 2*(m1*m1) = n*n by XCMPLX_1:5;
 then 2 divides n*n by NAT_1:def 3;
 then 2 divides n by INT_2:44,NEWTON:98;
 then consider n1 being Nat such that
W5: n=2*n1 by NAT_1:def 3;
A10: m1/n1 = sqrt 2 by W4,W5,XCMPLX_1:92,W2;
A11: n1>0 by W5,C,REAL_2:123;
  then 2*n1>1*n1 by REAL_2:199;
 hence contradiction by A10,W5,A11,W3;
end;
```

In Sectie 5 verderop zullen twee andere kleine Mizar-formalisaties in enig detail worden uitgelegd.

Mizar heeft de grootste bibliotheek van geformaliseerde stellingen van alle momenteel bestaande bewijsassistenten. De Mizar Mathematical Library (MML) bestaat uit 1.005 zogenaamde *artikelen*, formalisaties van een paar duizend regels per stuk, en is in totaal 68 megabyte ofwel 2,1 miljoen regels code groot, waarin in totaal 55.536 kleinere en grotere stellingen worden bewezen.

De webpagina van Mizar is:

`http://www.mizar.org/`

Mizar is door de bank genomen geen moeilijk systeem, maar het heeft een paar moeilijke kanten en er is helaas weinig documentatie, waardoor het toch niet aan te bevelen is om zonder hulp Mizar proberen te leren.

4 Geformaliseerde stellingen

Toen bewijsassistenten voor het eerst werden geïntroduceerd (door N.G. de Bruijn in Nederland dus) verwachten velen dat de technologie niet voor serieuze bewijzen bruikbaar zou zijn. In het begin van de twintigste eeuw hadden

Alfred North Whitehead en Bertrand Russell al geprobeerd bewijzen in volledige precisie op te schrijven in hun *Principia Mathematica*, en daarin was het pas op pagina 379 gelukt om de stelling

$$1 + 1 = 2$$

te bewijzen. Om deze reden ging men er aanvankelijk van uit dat het onpraktisch was serieuze wiskunde op een soortgelijke manier te behandelen.

Deze verwachting is niet bewaarheid: men had buiten de enorme toegevoegde waarde van het gebruik van computers gerekend. In de loop van de tijd zijn er meerdere zeer niet-triviale stellingen geformaliseerd. We bespreken nu de markantste voorbeelden.

De hoofdstelling van de algebra

Deze stelling was het onderwerp van het proefschrift van de 'Prins der Wiskundigen' Carl Friedrich Gauss. Dit proefschrift getiteld *Demonstratio nova theorematis omnem functionem algebraicam rationalem integram unius variabilis in factores reales primi vel secundi gradus resolvi posse*, ofwel *Een nieuw bewijs van de stelling dat iedere gehele rationale algebraïsche functie van één variabele in reële factoren van de eerste of tweede graad kan worden opgelost*, werd gepubliceerd in 1799. Gauss gaf in de loop van zijn leven vier bewijzen van de hoofdstelling van de algebra.

Carl Friedrich Gauss

De stelling zegt dat *iedere* niet-triviale vergelijking met alleen de basale rekenkundige operaties (optellen, aftrekken, vermenigvuldigen en delen) in de complexe getallen altijd een oplossing heeft. Voor de reële getallen geldt dit niet. Er is bijvoorbeeld geen reëel getal x waarvoor

$$x^2 + 1 = 0$$

Maar als we zo'n getal, $i = \sqrt{-1}$, toevoegen geldt dit wel, en geldt dit zelfs voor alle andere vergelijkingen, inclusief de vergelijkingen die we met deze extra getallen kunnen maken. Zo hoeven we bijvoorbeeld niet ook een wortel van i toe te voegen, want de vergelijking

$$x^2 - i = 0$$

heeft al de oplossingen

$$x = \frac{1}{2}\sqrt{2} + i\frac{1}{2}\sqrt{2} \quad \text{en} \quad x = -\frac{1}{2}\sqrt{2} - i\frac{1}{2}\sqrt{2}$$

De hoofdstelling van de algebra is in meerdere bewijsassistenten geformaliseerd. Er zijn formalisaties in Mizar, in HOL Light, in Coq (een intuïtionistisch bewijs, geformaliseerd aan de Radboud Universiteit Nijmegen) en in Isabelle.

De priemgetalstelling

We bedoelen met de priemgetalstelling *niet* de simpele stelling dat er oneindig veel priemgetallen zijn, maar de veel sterkere stelling dat de priemgetallen in de buurt van n ongeveer een dichtheid van $1/\ln(n)$ hebben. De precieze uitspraak van de priemgetalstelling is dat

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln(n)} = 1$$

waarbij $\pi(n)$ het aantal priemgetallen kleiner of gelijk n is. Om een indruk te geven van hoe goed deze limiet convergeert: voor n een biljoen (dus $n = 1.000.000.000.000$) geldt dat

$$\pi(n) = 37.607.912.018$$

(er zijn dus dit aantal priemgetallen kleiner dan een biljoen), terwijl

$$\frac{n}{\ln(n)} = 36.191.206.825,27\ldots$$

Een betere benadering krijg je overigens door de uitspraak over de dichtheid van $1/\ln(n)$ serieus te nemen en de *logaritmische integraal* uit te rekenen:

$$\mathrm{li}(n) = \int_0^n \frac{dt}{\ln(t)} = 37.607.950.280,80\ldots$$

(Het is makkelijk te bewijzen dat de priemgetalstelling equivalent is aan de stelling die zegt dat de limiet van de verhouding van $\pi(n)$ en $\mathrm{li}(n)$ ook naar één gaat.)

Het bewijs van deze stelling door Jacques Hadamard en Charles-Jean de la Vallée-Poussin, voortbouwend op de ideeën van Bernhard Riemann, was één van de hoogtepunten van de negentiende eeuwse wiskunde. Het maakt essentieel

gebruik van de analytische theorie over complexe functies, en gebruikt in het bijzonder eigenschappen van de verdeling van de complexe nulpunten van de Riemann zeta-functie

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

die we boven al hebben genoemd. Midden twintigste eeuw werd er door Atle Selberg en Paul Erdős ook een 'elementair' bewijs gevonden dat geen analyse nodig heeft.

Beide bewijzen van de priemgetalstelling zijn geformaliseerd. Het analytische bewijs uit de negentiende eeuw die een heleboel complexe functietheorie nodig heeft is zeer recent geformaliseerd door John Harrison in HOL Light, en het elementaire bewijs was al eerder geformaliseerd in Isabelle door een groepje mensen onder leiding van Jeremy Avigad van de Carnegie Mellon universiteit.

De stelling over Jordan-krommen

Deze stelling is interessant omdat het een heel eenvoudige uitspraak betreft die als je hem probeert te bewijzen heel erg moeilijk blijkt. Hij werd bewezen door Oswald Veblen in 1905, en leidde tot het vakgebied van de *topologie*, waarvan bovengenoemde L.E.J. Brouwer één van de grondleggers is geweest. De stelling zegt dat iedere *Jordan kromme* (een continue gesloten kromme in het platte vlak die zichzelf niet doorsnijdt: een gesloten lus dus) het vlak in precies twee delen verdeelt, een eindige binnenkant en een oneindige buitenkant, waarbij de kromme de rand van beide delen is. Intuïtief lijkt het alsof hier niets te bewijzen is ('dat zie je toch zo'), maar als je het wiskundig precies maakt is dit dus een moeilijk resultaat.

In het Mizar project is er jaren gewerkt aan de formalisatie van een bewijs van deze stelling, vooral omdat er voor een onhandig bewijs was gekozen. Inmiddels zijn er twee formalisaties, één in HOL Light door Tom Hales, en één in Mizar door een groep mensen onder leiding van Andrzej Trybulec, waarbij het bewijs uiteindelijk werd voltooid door Artur Korniłowicz.

De onvolledigheidsstelling van Gödel

Dit is waarschijnlijk de beroemdste stelling van de twintigste eeuw, bewezen door Kurt Gödel in 1931. De stelling komt erop neer dat voor *iedere* enigszins redelijke collectie axioma's er ware uitspraken bestaan die niet uit die axioma's bewezen kunnen worden. (Om het interessant te maken heeft Gödel ook een *volledigheidsstelling* – die ook heel belangrijk is, en trouwens ook geformaliseerd – maar de onvolledigheidsstelling is veel schokkender.)

In feite zijn er *twee* onvolledigheidsstellingen, de eerste en de tweede. De tweede onvolledigheidsstelling voegt aan de eerste toe dat je uit een *consistent* stel axioma's – dat betekent dat je met de axioma's geen onwaarheden kan bewijzen – nooit kan bewijzen dat die axioma's inderdaad consistent zijn. Een systeem kan zijn eigen redelijkheid niet inzien, zeg maar.

De eerste onvolledigheidsstelling is een aantal keer geformaliseerd: in een voorganger van ACL2 door Natarajan Shankar (de maker van de PVS bewijsassistent), in Coq door Russell O'Connor, en in HOL Light door John Harrison. De tweede onvolledigheidsstelling is tot vandaag niet geformaliseerd, hoewel er wel een aantal mensen mee bezig is.

De vierkleurenstelling

Deze stelling zegt dat je bij een kaart met een aantal landen altijd ieder land één van vier kleuren kan geven op zo'n manier dat er dan geen twee landen met dezelfde kleur aan elkaar grenzen.

Deze stelling was lange tijd een open probleem, en werd pas bewezen in 1975 door Kenneth Appel en Wolfgang Haken. Daarbij gebruikten ze een heleboel computertijd om ontzettend veel kleuringen van 1.936 specifieke kaarten te analyseren. Om deze reden was hun bewijs omstreden, aangezien het niet mogelijk was het bewijs te geloven zonder een computer te hoeven vertrouwen.

In 1996 werd het bewijs nog eens dunnetjes overgedaan door Neil Robertson, Daniel P. Sanders, Paul Seymour en Robin Thomas, waarbij het werd teruggebracht tot een veel overzichtelijker vorm. Het bewijs en de bijbehorende computerprogramma's waren nu nog maar enige tientallen pagina's lang, en er waren nog maar 633 in plaats van 1.936 kaarten die moesten worden geanalyseerd.

Georges Gonthier

In 2004 werd de vierkleurenstelling in Coq geformaliseerd door Georges Gonthier, met de hulp van Benjamin Werner. Dit is één van de meest indrukwekkende formalisaties tot nog toe. In deze formalisatie werden zowel de computerprogramma's uit het bewijs correct bewezen, als de hele topologische theorie die bij het probleem hoort geformaliseerd.

Voor dit project werd door Georges Gonthier speciaal een nieuwe bewijstaal voor Coq ontwikkeld met de naam *ssreflect*. Deze taal geldt als een enorme

stap voorwaarts in hoe efficiënt met het Coq systeem bewijzen kunnen worden geformaliseerd. Helaas is er momenteel nog geen goede documentatie van deze taal beschikbaar, en is deze dus nog moeilijk te leren.

Georges Gonthier werkt voor Microsoft research, en er is een instituut opgericht als samenwerking tussen Microsoft en INRIA, waarin hij momenteel met een groepje mensen de moeilijke Feit-Thompson stelling uit de groepentheorie aan het formaliseren is. Dit als mogelijke aanloop naar een toekomstige formalisatie van de beruchte stelling over de classificatie van de eindige groepen.

5 Voorbeelden van formalisaties

We zullen nu twee eenvoudige voorbeelden van een formalisatie laten zien. Omdat het het makkelijkst is om Mizar formalisaties te begrijpen (hoewel het helaas niet zo makkelijk is om Mizar te leren schrijven) hebben we hier gekozen voor voorbeelden in Mizar.

Geen grootste getal

Het eerste voorbeeld is een formalisatie van een vrolijk rijmpje van Marjolein Kool dat ik vond op het weblog van de wiskundemeisjes, Ionica Smeets en Jeanine Daems:

`http://www.wiskundemeisjes.nl/`

Het gaat om een bewijs van de volkomen triviale stelling dat er geen grootste getal bestaat. We geven nu het rijmpje, met daarnaast hetzelfde bewijs in de Mizar bewijstaal:

Een bolleboos riep laatst met zwier
gewapend met een vel A-vijf:
Er is geen allergrootst getal,
dat is wat ik bewijzen ga.
Stel, dat ik u nu zou bedriegen
en hier een potje stond te jokken,
dan ik zou zonder overdrijven
het grootste kunnen op gaan noemen.
Maar ben ik klaar, roept u gemeen:
'Vermeerder dat getal met twee!'
En zien we zeker en gewis
dat dit toch niet het grootste was.
En gaan we zo nog door een poos,
dan merkt u: dit is onbegrensd.
En daarmee heb ik q.e.d.
Ik ben hier diep gelukkig door.
'Zo gaan', zei hij voor hij bezwijmde,
'bewijzen uit het ongedichte'.

```
theorem
  not ex n st for m holds n >= m
proof

  assume not thesis;
  then consider n such that
    for m holds n >= m;

  set n' = n + 2;

  n' > n;

  then not for m holds n >= m;

  hence contradiction;

end;
```

Mizar vindt deze vertaling helaas niet acceptabel. Het systeem kan namelijk niet uit zichzelf begrijpen waarom de twee onderstreepte regels volgen. In het bijzonder moeten we het systeem vertellen dat de contradictie volgt met de eigenschap van n, en ook is er een lemma nodig met de naam `XREAL_1:31`

```
0 < a implies b < b+a
```

om te bewijzen dat $n' > n$. Als we zo deze formalisatie afmaken wordt het:

```
theorem
  not ex n st for m holds n >= m
proof
  assume not thesis;
  then consider n such that
A1: for m holds n >= m;
  set n' = n + 2;
  n' > n by XREAL_1:31;
  then not for m holds n >= m;
  hence contradiction by A1;
end;
```

Overigens kan Mizar, als we niet het rijmpje letterlijk hoeven te volgen, deze stelling ook wel sneller bewijzen:

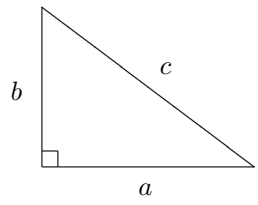
```
theorem
  not ex n st for m holds n >= m
proof
  let n;
  n + 2 > n by XREAL_1:31;
  hence thesis;
end;
```

Een formule voor Pythagoreïsche drietallen

Een *Pythagoreïsche drietal* is een drietal positief gehele getallen a, b en c met

$$a^2 + b^2 = c^2$$

Het gaat hier dus met de stelling van Pythagoras om rechthoekige driehoeken



waarvan de lengtes van de zijden alledrie een geheel getal zijn. Nu is er een aardige formule – die al in Euclides' *De Elementen* voorkomt – om dit soort Pythagoreïsche drietallen te genereren:

$$\begin{aligned} a &= m^2 - n^2 \\ b &= 2mn \\ c &= m^2 + n^2 \end{aligned}$$

We krijgen met deze formule bijvoorbeeld de volgende voorbeelden:

$$\begin{array}{llll} m=2 & n=1 & \longrightarrow & 3^2 + 4^2 = 5^2 \\ m=3 & n=2 & \longrightarrow & 5^2 + 12^2 = 13^2 \\ m=4 & n=1 & \longrightarrow & 15^2 + 8^2 = 17^2 \\ m=4 & n=3 & \longrightarrow & 7^2 + 24^2 = 25^2 \end{array}$$

Je kunt je nu de vraag stellen of je op deze manier *alle* Pythagoreïsche drietallen krijgt, en het antwoord is: bijna. Je krijgt ze niet allemaal – zo krijg je bijvoorbeeld niet $9^2 + 12^2 = 15^2$, want 15 is niet de som van twee kwadraten – maar je krijgt wel alle *vereenvoudigde* Pythagoreïsche drietallen, waarbij het drietal niet een veelvoud is van een ander Pythagoreïsch drietal.

Het bewijs van dit feit is niet heel ingewikkeld, en we geven het hier in Mizar syntax:

```
let a,b,c; assume a^2 + b^2 = c^2;
assume a,b are_relative_prime;
then a is odd or b is odd; assume a is odd;

ex m,n st a = m^2 - n^2 & b = 2*m*n & c = m^2 + n^2
proof
  b is even; c is odd;
X: (c + a)/2,(c - a)/2 are_relative_prime;
  ((c + a)/2)*((c - a)/2) = (c^2 - a^2)/4 .= (b/2)^2;
  then ((c + a)/2)*((c - a)/2) is square;
  then (c + a)/2 is square & (c - a)/2 is square by X;
  consider m,n such that m^2 = (c + a)/2 & n^2 = (c - a)/2;
  take m,n;
  thus a = (c + a)/2 - (c - a)/2 .= m^2 - n^2;
  b^2 = (c + a)*(c - a) .= 4*m^2*n^2 .= (2*m*n)^2;
  hence b = 2*m*n;
  thus c = (c + a)/2 + (c - a)/2 .= m^2 + n^2;
end;
```

De clou van dit bewijs is dat geldt dat

$$\Big(\frac{c+a}{2}\Big)\Big(\frac{c-a}{2}\Big) = \Big(\frac{b}{2}\Big)^2$$

een kwadraat is, terwijl de factoren $(c+a)/2$ en $(c-a)/2$ geen delers gemeenschappelijk hebben (dit zijn regels 2 en 4 in de Mizar-versie.) Daaruit volgt dat

beide factoren ook een kwadraat moeten zijn (regel 5), en daarmee is het bewijs al bijna klaar.

Helaas is, net als bij het rijmpje van het grootste getal, bovenstaande Mizar tekst te kort door de bocht om door Mizar geaccepteerd te worden. Bij bijna iedere regel klaagt het systeem dat het niet begrijpt waarom dit volgt. Om het af te maken is er nog veel werk nodig: er moeten labels, verwijzingen naar lemmas, en extra stapjes worden toegevoegd. Het eindresultaat is daardoor helaas veel minder leesbaar dan bovenstaande versie. Om hier een indruk van te geven, de regels in bovenstaande tekst:

```
X: (c + a)/2,(c - a)/2 are_relative_prime;
  ((c + a)/2)*((c - a)/2) = (c^2 - a^2)/4 .= (b/2)^2;
  then ((c + a)/2)*((c - a)/2) is square;
  then (c + a)/2 is square & (c - a)/2 is square by X;
  consider m,n such that m^2 = (c + a)/2 & n^2 = (c - a)/2;
```

worden in de uiteindelijk Mizar-formalisatie die je krijgt door het verder uit te werken opdat het Mizar systeem het accepteert:

```
X: (c + a)/2,(c - a)/2 are_relative_prime by Lm3;
  ((c + a)/2)*((c - a)/2) = ((c + a)*(c - a))/(2*2) by REAL_1:35
    .= (c^2 - a^2)/4 by SQUARE_1:67
    .= (b^2)/(2*2) by H1,INT_1:3
    .= (b^2)/(2^2) by SQUARE_1:def 3
    .= (b/2)^2 by SQUARE_1:69;
  then ((c + a)/2)*((c - a)/2) is square by A1,Def1;
  then (c + a)/2 is square & (c - a)/2 is square by X,Lm4;
  then (ex m st m^2 = (c + a)/2) &
    (ex n st n^2 = (c - a)/2) by Def1;
  then consider m,n such that
A9: m^2 = (c + a)/2 & n^2 = (c - a)/2;
```

Hierin verwijzen `A1`, `Lm3`, `Lm4` en `Def1` naar labels elders in de formalisatie. Het lemma `Lm4` luidt bijvoorbeeld:

```
x,y are_relative_prime & x*y is square implies x is square & y is square
```

De volledige uitgewerkte formalisatie van het bewijs is uiteindelijk 97 in plaats van 11 regels lang. (En dat is exclusief de bewijzen van de hulplemma's.)

6 De drie revoluties in de wiskunde

In de geschiedenis van de wiskunde zijn drie revoluties geweest:

Bewijzen

De eerste revolutie was de ontwikkeling van *bewijzen* in de Griekse oudheid. Vóór deze revolutie bestond wiskunde voornamelijk uit *berekeningen*. Deze Griekse ontwikkeling vond zijn hoogtepunt in *De Elementen* van Euclides, een boek waarin – bewijs na bewijs – de meetkunde systematisch werd ontwikkeld.

Euclides van Alexandrië

Rigor

De tweede revolutie was de ontwikkeling van *rigor* aan het eind van de negentiende eeuw. Daarvóór was wiskunde niet volledig precies. Zo rammelt Euclides' ontwikkeling van de meetkunde als je er met moderne ogen naar kijkt, en ook de ontwikkeling van de infinitesimaalrekening door Isaac Newton en Gottfried Leibniz was niet rigoreus, met referenties naar oneindig kleine grootheden. Eind negentiende eeuw kwam hieraan een eind met de ontwikkeling van ϵ/δ definities van limieten door Augustin Louis Cauchy. Deze ontwikkeling culmineerde in de verzamelingenleer van Georg Cantor en de ontwikkeling van de mathematische logica door Gottlob Frege.

De meetkunde van Euclides werd voor het eerst rigoreus gemaakt rond de eeuwwisseling door David Hilbert. Een bijzonder fraaie variant hierop werd later ontwikkeld door Alfred Tarski.

Formalisatie

De derde revolutie is de ontwikkeling van praktische *formele* wiskunde. Hierbij wordt wiskunde in de computer gerepresenteerd op een manier dat *alle* details in de computer aanwezig zijn. De computer kan hierdoor de correctheid van de wiskunde volledig mechanisch nagaan. Bij rigoreuze wiskunde is het *in principe* mogelijk om bewijzen volledig precies op te schrijven, maar bij formele wiskunde wordt dit ook *in de praktijk* gedaan. Deze derde revolutie is momenteel in volle gang.