Roel de Vrijer
Vrije Universiteit Amsterdam
Afdeling Informatica
r.c.de.vrijer@vu.nl

In Memoriam Nicolaas Govert de Bruijn (1918–2012)

Wiskunde als taal: het Automath-project

Met het Automath-project ontwikkelde Dick de Bruijn een computertaal voor het verifiëren van wiskundige bewijzen. Roel de Vrijer, onderzoeker aan de Vrije Universiteit en oud-medewerker van het Automath-project, beschrijft hoe Automath opgebouwd is en wat de betekenis ervan geweest is.

In juni 1970 gaf professor De Bruijn uit Eindhoven op uitnodiging van de Utrechtse vereniging van wis- en natuurkundestudenten voorkandidaten A-E een lezing over Automath. Ik heb de korte aankondigingstekst, die De Bruijn vooraf aan ons had opgestuurd, bewaard. Zie kader.

Het idee wiskundige bewijzen te laten verifiëren door een computer was nog nieuw en ongehoord. Ik herinner me hoe hij op geamuseerde toon vertelde dat er standaard twee manieren waren waarop logici op zijn plannen met Automath plachten te reageren. De ene groep zei: maar Gödel heeft in 1931 toch bewezen dat wat je voorstelt niet mogelijk is? En de andere: maar dat hebben Russell en Whitehead in 1910 toch allang gedaan? Ik heb hem deze anekdote ook later nog vele malen horen reciteren.

In de logische kringen waarin ik korte tijd later kwam te verkeren, werd inderdaad met scepsis tegen de onderneming aangekeken. Als het al mogelijk was, vroeg men zich af, wat droeg het dan bij aan de fundamentele vragen van de grondslagen van de wiskunde? Ik kan dat minder goed beoordelen, maar mijn indruk is dat vanuit wiskundige hoek de reacties in die begintijd niet veel bemoedigender waren.

Hoe dit ook zij, De Bruijn geloofde in Automath en deed er alles aan het tot een succes te maken. De twee medewerkers die hij in 1967 en 1969 aan kon stellen aan de TH Eindhoven liet hij allebei aan Automath werken en daarnaast bereidde hij een groot onderzoeksproject voor, dat vanaf 1972 voor vijf jaar door ZWO werd gefinancierd. Toen ikzelf in 1972 afstudeerde, was er nog ruimte in het Automath-project, en dat werd mijn eerste baan. Ik kwam terecht in de hechte groep mensen die De Bruijn om zich heen had verzameld. We hadden allemaal het gevoel met iets belangrijks en baanbrekends bezig te zijn dat in de rest van de wereld nog niet als zodanig werd herkend. Er was een veelheid aan taken. In de Automath-groep werd geprogrammeerd aan de checker, er werd wiskunde in Automath vertaald of geschreven, er werd gewerkt aan het formalisme zelf, dat noemden we de taaltheorie, en er was een secretaresse die de met de hand geschreven Automath-teksten omzette in computerbestanden. Op het hoogtepunt, in 1975, bestond de Automath-groep uit een tiental personen.

Er waren regelmatig werkbijeenkomsten. Bij die gelegenheden, maar ook als je zomaar eens bij hem binnenliep, kon De Bruijn met een enorm enthousiasme over een nieuw idee vertellen. Bijvoorbeeld hoe je een heel wiskundeboek, of zelfs de hele wiskunde, kon voorstellen als een enkele term, die hij dan tekende als een liggende boom, met heel veel takjes. Hij werkte zelf ook met onvermoeibare vlijt door aan Automath. In de kwarteeuw tussen 1967 en 1993 produceerde hij een gestage reeks artikelen, rapporten en notities.

Het Automath-project heeft zelf niet direct voor de doorbraak gezorgd waar misschien op gehoopt werd, maar de betekenis ervan is enorm geweest. Bewijsverificatie, proof assistants, propositions-as-types, typentheorie met dependent types, logical

Aankondigingstekst lezing De Bruijn

AUTOMATH werd sinds 1967 ontwikkeld aan de TH Eindhoven. Het is een taal die voldoet (of tracht te voldoen) aan:

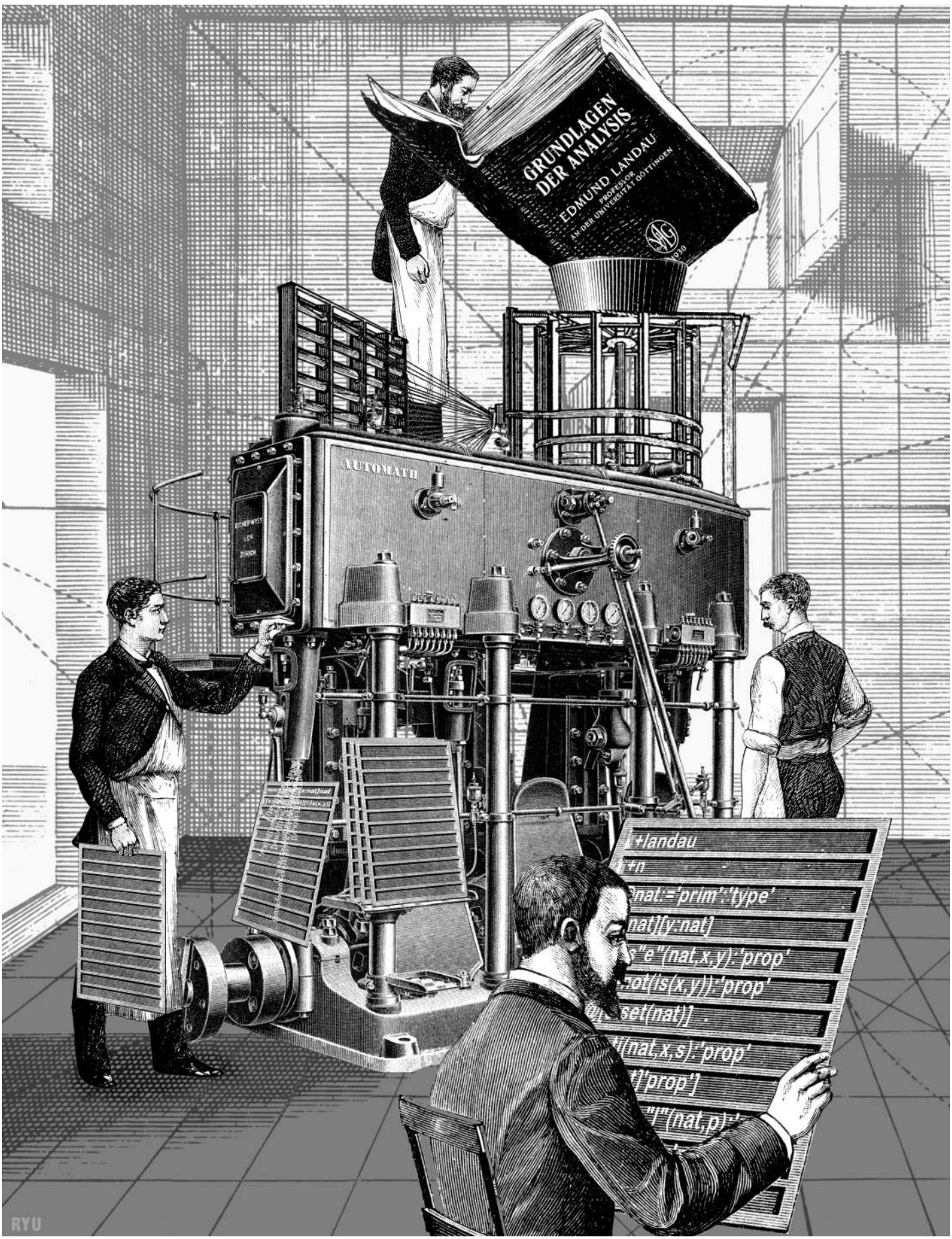
1. Elk wiskundig betoog, over welk onderwerp ook, kan erin worden uitgedrukt.
2. Alles wat er op concrete wijze in gezegd wordt, is wiskundig correct.
3. Er is een programma dat een computer in staat stelt een hem aangeboden tekst op correctheid te onderzoeken.

AUTOMATH doet in zijn peuterige detaillering denken aan de machinecode van een computer. Men kan proberen supertalen te maken die zich gemakkelijker laten schrijven en die zich machinaal in AUTOMATH laten omzetten.

Men kan zich desgewenst voorstellen dat een in AUTOMATH geschreven boek de gehele wiskunde omvat. In iedere regel van zo'n boek mag van alle voorgaande regels gebruik gemaakt worden.

De regels hebben alle de volgende vorm: in de context (1) is (2) een door (3) gegeven ding van de soort (4). Hierin is (2) een nieuwe naam (identificator).

Ogenschijnlijk is dit soort regels slechts voor het geven van definities bruikbaar, maar door het gebruik van assertiesoorten op de plaats (4) kunnen evengoed bewijzen worden geschreven.



Illustratie: Ryu Tajiri

frameworks, calculi van expliciete substituties..., het hoort bij de belangrijkste ontwikkelingen in de logica in de tweede helft van de twintigste eeuw, en het begon allemaal bij Automath. De lijst van wiskundige resultaten die inmiddels zijn geformaliseerd en geverifieerd met *proof assistants* ontwikkeld in de voetsporen van Automath, zoals Coq, NUPRL, HOL, HOL-light, Isabelle..., is indrukwekkend. Er zijn zware en tot de verbeelding sprekende stellingen bij, zoals de Hoofdstelling van de Algebra, de Eerste Onvolledigheidsstelling van Gödel, de Vierkleurenstelling en, zeer recent, de Stelling van Feit en Thompson over de oplosbaarheid van eindige groepen van oneven orde. Proof assistants worden tegenwoordig ook ingezet bij de verificatie en *certificering* van software en hardware. Een sprekend voorbeeld is de ARM6-processor, waarvan de specificaties zijn geverifieerd met behulp van de proof assistant HOL.

De meest in het oog springende doelstelling van Automath is natuurlijk *proof checking*, en daarin zijn ook de meest spectaculaire resultaten geboekt. Maar De Bruijn had een drijfveer die voor hem even belangrijk was. Het heeft te maken met de vraag wat wiskunde is. Hij wilde de structuur doorgronden van de wiskunde zoals die zich manifesteert, en dat is in de vorm van wiskundige teksten. De Bruijn ontdekte dat dit mogelijk is met een zeer beperkt arsenaal van bouwstenen. Deze bouwstenen zijn altijd dezelfde,

De Bruijn in 1966 uitkijkend over Eindhoven op het dak van de THE, ten tijde van zijn keuze om in Eindhoven te blijven

in welk gebied van de wiskunde je je ook begeeft, en of het nu wiskunde is of logica.

Ik wil hier nu graag iets meer laten zien van Automath, en daaromheen nog het een en ander vertellen over uitgangspunten, doelstellingen en resultaten. Daarvoor is het nuttig straks ook even af te dalen naar het niveau van 'peuterige detaillering' waarvan sprake is in het bovengeciteerde convocaat voor A-E.

Aspecten van Automath

Zoals gezegd berustte het rotsvaste vertrouwen van De Bruijn dat alle wiskunde in Automath uitgedrukt kan worden op de overtuiging dat hij de principes waarop een wiskundig betoog berust in kaart had gebracht. Ze zijn dezelfde voor elke tak van wiskunde en neutraal tegenover de logische grondslagen. De specifieke kenmerken van een wiskundige of logische benadering kan de gebruiker zelf in zijn formalisering in Automath aanbrengen.

De Bruijn heeft er altijd over gewaakt die neutraliteit te behouden en geen specifieke wiskundige constructies in het framework op te nemen. In dat opzicht verschilt Automath bijvoorbeeld wezenlijk van het systeem Coq, dat op Automath geënt is. In Coq is een faciliteit ingebouwd om automatisch inductieve datatypen, zoals naturals of strings, te genereren. In 1991 heeft de Bruijn zijn standpunt nog eens uiteengezet in een artikel met de veelzeggende titel 'A plea for weaker frameworks' [4].

Dat alle wiskunde in Automath kan worden uitgedrukt is vooral ook een praktische claim. Om theoretische uitdrukkingskracht was het De Bruijn niet te doen. Het vertalen van wiskunde moest uitvoerbaar zijn. Om dat te testen en te demonstreren werd al kort na de aanvang van het project door De Bruijns medewerker Bert van Benthem Jutting begonnen met de vertaling van het boek *Grundlagen der Analysis* van Edmund Landau [7] in Automath. Het werd een succes, in 1975 was de vertaling een feit en was het geheel door de machine gecheckt.

De ruim 130 pagina's Duitse wiskundetekst van Landau waren door Jutting vertaald in 13433 regels Automath, ongeveer 100 regels per pagina. De Bruijn had voorspeld dat de verhouding tussen de omvang van vertaalde en oorspronkelijke tekst min of meer constant zou blijven, dus bijvoorbeeld in hoofdstuk 5 van Landau niet wezenlijk verschillend van hoofdstuk 2. Zijn meer algemene hypothese was dat deze factor, de *De Bruijn-factor*, ook niet zou toenemen bij het verder binnendringen van een geavanceerd wiskundig specialisme. Het gevaar van exponentiële groei van de benodigde inspanning wordt daardoor uitgesloten. Zowel bij de Landau-vertaling als bij omvangrijke formaliseringen in later ontwikkelde systemen voor proof checking, is De Bruijns hypothese steeds bevestigd. Hoe groot de De Bruijn-factor precies is, hangt af van hoe je meet, en ook van de mate van gedetailleerdheid van de oorspronkelijke tekst. Bij de Landau-vertaling kwam de De Bruijn-factor volgens de berekening van Van Benthem Jutting uit op ongeveer 5 à 6. Voor de respectievelijke hoofdstukken 1 tot 5 toonde de ratio het volgende verloop: 3,8 5,3 5,7 6,4 5,5. Wiedijk [9] komt met een andere berekeningsmethode op een De Bruijn-factor van 3,7 voor de Landau-vertaling.

We keren even terug naar het A-E-convocaat. Volgens punt 2 dient correct Automath garant te staan voor correcte wiskunde. Het vertrouwen daarin kan alleen maar gebaseerd zijn op de transparantie en natuurlijkheid van de taaldefinitie die bepaalt wat een correct Automath-boek is. Dit levert dan ook meteen een bijkomend argument voor 'weaker frameworks'. In praktische zin kun je de vraag naar correctheid niet los zien van het computerprogramma dat voor de verificatie wordt gebruikt. De Bruijn zei hierover in [5] het volgende: "If we work on the principle that a verification program provides the only norm for correctness, it becomes a social problem to agree on a particular system. Such an agreement can only be reached if the system is exceedingly simple and transparent." Het programma van een proof checker moet overzichtelijk en transparant zijn, en liefst klein van formaat. Dit beginsel wordt tegenwoordig het *De Bruijn-criterium* genoemd.

De Bruijn heeft zelf de eerste checker voor Automath geschreven. Volgens eigen zeggen was een motief om met Automath te beginnen ook dat hij een interessante en originele uitdaging zocht om te programmeren. In de jaren van het Automath-project is er een nieuwe checker ontwikkeld voor de toen op de TH Eindhoven aanwezige Burroughs B7600 computer. Die gebruikte de algoritmes van De Bruijn, de problemen die moesten worden opgelost hadden voor een belangrijk deel te maken met zaken als input/output en geheugenallocatie. Met dat programma is in 1975 de Landau-vertaling gecheckt, met een laatste doorloop op 18 oktober 1975. Er is geen op huidige machines executeerbare code van overgebleven.

Een nieuw programma waarmee oorspronkelijke Automath-teksten kunnen worden gecheckt, is geschreven door Freek Wiedijk [9] en is nog steeds beschikbaar. De Landau-

Edmund Landau's boek *Grundlagen der Analysis* wordt door Automath gecheckt

vertaling van Jutting is er opnieuw mee geverifieerd.

Ik noem nog een paar aspecten van Automath die hier vanwege ruimtegebrek niet verder aan bod komen. In het A-E-convocaat heeft De Bruijn het al over de supertalen die zich gemakkelijker laten schrijven en die zich machinaal in Automath laten omzetten. Dit kan als een vooraankondiging worden beschouwd van het project Wiskundige Omgangstaal (WOT), waar de De Bruijn vanaf de latere jaren zeventig veel energie in heeft gestoken, in samenwerking met zijn medewerkers en met collega's uit de didactiek van de wiskunde. Naast het element van bewijsverificatie biedt een volledige formalisering van wiskunde, zoals in Automath beoogd, nog meer perspectieven. Bijvoorbeeld het aanleggen van een publieke database van gecheckte wiskunde, die iedereen kan gebruiken en waar iedereen op voort kan bouwen. Of de mogelijkheid met een druk op de knop te zien wat wel en wat niet relevant was om te komen tot een wiskundig resultaat (excerpten).

Dan nu de bovengenoemde regels van de vorm: in de context (1) is (2) een door (3) gegeven ding van de soort (4). Die vormen het eigenlijke Automath.

Boeken en lijnen

Een tekst in Automath bestaat uit opeenvolgende regels. De tekst wordt *boek* genoemd, en de regels heten *lijnen*. Een lijn heeft de vorm $c * b := P : Q$, waarin b een nieuwe naam is en Q een type, het *type* van b. De c is een context-indicator. Bij types kun je in eerste instantie denken aan substantieven zoals natuurlijk getal of complexe functie. Voor de invulling van P zijn er drie mogelijkheden.

- P is de tekst PN (voor *primitive notion*). Dit duidt aan dat het bestaan van b wordt gepostuleerd, zoals in de mededeling "1 is een natuurlijk getal" bij de introductie van de natuurlijke getallen.
- P is niet ingevuld, aangeduid door —. De identifier b wordt dan opgevoerd als een variabele, of, in Automath-jargon, een *block opener*.
- P is een expressie, die dan opgevat moet worden als een definitie van b. Voor de correctheid van een definitielijn is het vereist dat op basis van het voorafgaande kan worden afgeleid dat de definiërende expressie P het in de rechterkolom aangeduide type Q heeft.

	$*$	*nat*	:=	PN	type
	$*$	1	:=	PN	*nat*
	$*$	n	:=	—	*nat*
n	$*$	*suc*	:=	PN	*nat*
n	$*$	*plus2*	:=	*suc*(*suc*(n)	*nat*
	$*$	2	:=	*suc*(1)	*nat*
	$*$	3	:=	*plus2*(1)	*nat*

Figuur 1 Kort boek met alle drie soorten lijnen

In Figuur 1 is een kort boek te zien waarin we alle drie soorten lijnen aantreffen. Dit fragment zou een formalisering kunnen zijn van een wiskundige tekst in de trant van: "We introduceren de verzameling natuurlijke getallen. 1 is een natuurlijk getal. Laat n een natuurlijk getal zijn. Dan is *suc*(n) ook een natuurlijk getal. We definiëren de functie *plus2* door *plus2*(n) = *suc*(*suc*(n)). Tenslotte definiëren we 2 en 3 als de natuurlijke getallen *suc*(1) en *plus2*(1)."

Bovenstaande zeven lijnen vormen een correct boek in PAL, wat een afkorting is voor *Primitive Automath Language*. PAL is een deeltaal van Automath, elk correct PAL-boek is ook een correct Automath-boek. De Bruijn benadrukte dat PAL volstaat voor de formalisering van de meeste wiskunde tot ver in de negentiende eeuw, totdat het door Euler geïnitieerde moderne functiebegrip gemeengoed was geworden.

Bewijzen als objecten

Het fundamentele inzicht van De Bruijn waar Automath op berust, is dat het getoonde mechanisme voor het postuleren, veronderstellen en definiëren van types en objecten ook van toepassing is op beweringen en bewijzen. Daarbij worden bewijzen opgevat en behandeld als wiskundige objecten. Bij een propositie ϕ hoort dan het type van ϕ's bewijzen, zeg Pr_ϕ. Een bewijs geven van ϕ komt in Automath neer op het construeren van een object met type Pr_ϕ.

Er waren drie mogelijke invullingen van P in een lijn $c * b := P : Q$. Wanneer Q nu een bewijstype Pr_ϕ is voor de propositie ϕ, komen die hier op neer:

- P is de tekst PN. Dan wordt b als een bewijs van ϕ gepostuleerd. Dit stemt overeen met het aannemen van ϕ als een axioma.
- P is een block opener —. Een b in Pr_ϕ wordt veronderstellenderwijs ingevoerd. Dit komt overeen met een assumptie: "Stel dat ϕ".
- P is een expressie die b definieert, en waarvan kan worden geverifieerd dat het type inderdaad Pr_ϕ is. Dan is P, en daarmee ook b, een bewijs van ϕ.

	$*$	α	:=	—	type
α	$*$	x	:=	—	α
x	$*$	y	:=	—	α
y	$*$	IS	:=	PN	type
x	$*$	refl	:=	PN	IS(x, x)
	$*$	*stelling1*	:=	refl(*nat*, 1, 1)	IS(*nat*, 1, 1)

Figuur 2 Gelijkheid IS(α, x, y) in *prop*-stijl geïntroduceerd

Blijft nog de vraag hoe de propositie ϕ zelf in Automath gerepresenteerd word, en hoe de link van ϕ met Pr_ϕ tot stand komt. Hiervoor heeft De Bruijn meteen al bij de introductie van Automath twee mogelijkheden geopperd, die beide werken en zijn toegepast. De eerste wordt *bool*-stijl genoemd, de tweede *prop*-stijl.

- *bool*-stijl: er is een type *bool* van proposities, waar dus ook ϕ als object in leeft. Met de PAL-methode kan men dan een functie *Pr* postuleren die aan elke ϕ in *bool* een bijbehorend bewijstype *Pr*(ϕ) toekent.
- *prop*-stijl: het bewijstype Pr_ϕ wordt geïdentificeerd met de propositie ϕ zelf. (Deze oplossing is, hoofdzakelijk buiten Automath om, bekend geworden onder de naam *propositions as types*. Zie ook de volgende paragraaf.)

In de lijnen in Figuur 2 wordt in *prop*-stijl gelijkheid IS(α, x, y) van x en y op een type α geïntroduceerd, met een axioma voor reflexiviteit, en bewezen dat $1 = 1$.

λ-calculus

In het PAL-boek is een impliciete, contextuele, definitie van de functie *plus2* gegeven. Maar we kunnen nog geen operaties op functies definiëren, zoals bijvoorbeeld compositie.

Automath ontstaat door aan PAL een mechanisme van expliciete functiedefinitie toe te voegen, in de vorm van λ-abstractie:

$$plus2fun = \lambda x : nat\,.\,suc(suc(x))$$

De Bruijn had met deze notatie kennisgemaakt via Freudenthal en was deze zelf gaan toepassen in zijn colleges over functionaalanalyse. In [3] breekt hij een lans voor het gebruik van de λ-notatie in de gewone wiskundige praktijk. En hij gaat nog iets verder: "Het is me nooit duidelijk geworden waarom de lambda-notatie niet al een dikke honderd jaar eerder in zwang is gekomen,

	∗	*plus2fun*	:=	$[x : nat]\, suc(suc(x))$	$[x : nat]nat$
	∗	*3alt*	:=	$\langle 1\rangle$*plus2fun*	*nat*
	∗	f	:=	—	$[x : nat]nat$
f	∗	g	:=	—	$[x : nat]\, nat$
g	∗	*comp*	:=	$[x : nat]\,\langle\langle x\rangle f\rangle g$	$[x : nat]\, nat$
	∗	*plus4fun*	:=	*comp*(*plus2fun*, *plus2fun*)	$[x : nat]nat$

Figuur 3 Uitbreiding PAL-boek uit Figuur 1

en nog minder waarom de overgrote meerderheid van de analytici nog steeds zonder lambda's leeft. Eigenlijk is dat een historische schande." Freudenthal gebruikte geen λ, maar een eigen symbool Ψ, en De Bruijn introduceerde in Automath ook een nieuwe notatie: $[x : nat]\, suc(suc(x))$. Functietoepassing wordt in Automath gerepresenteerd door het argument vóór de functie te plaatsen, tussen speciale haakjes, zoals in de expressie $\langle 1\rangle[x : nat]\, suc(suc(x))$, die in één zogenaamde β-stap reduceert naar $suc(suc(1))$.

Het was typerend voor De Bruijn dat hij met zijn notatie afweek van wat gebruikelijk was in de λ-calculus en daar ondanks tegenwerpingen in volhardde. Hij vond zijn notatie superieur, en daar had hij ook gelijk in. In de gebruikelijke notatie, $(\lambda x : nat\,.\, suc(suc(x)))1$, kan er enig speurwerk nodig zijn om een corresponderend argument bij een λ te zoeken, in De Bruijns notatie staat het er gewoon naast. Hij vergeleek de traditionele notatie graag met een stoomtrein met aan de voorkant de locomotief en de steenkolenwagon helemaal achteraan. De machinist moet telkens de kolen over de hele lengte van de trein van achteren naar voren sjouwen. Hoeveel makkelijker is het niet met de kolenwagon direct aan de locomotief gekoppeld, zoals in Automath.

Om te laten zien hoe de λ-calculus functioneert in Automath, breiden we het eerdere PAL-boek uit met een paar extra lijnen. Zie Figuur 3. Daarin wordt een alternatieve variant *plus2fun* van *plus2* gedefinieerd, nu door λ-abstractie, en een definitie gegeven van *plus4fun* via *comp*, compositie van functies van natuurlijke getallen.

De type-aanduiding $[x : nat]$ nat kan gelezen worden als het type $nat \to nat$, van de functies van *nat* naar *nat*. De reden voor deze notatie is dat hij ook geschikt is voor de weergave van zogenaamde *dependent types*, zie de volgende paragraaf. Er zijn dus twee manieren om functies te introduceren in Automath, contextueel, zoals in PAL, en expliciet met een λ. Daardoor ontstaan keuzes. Zo zou je voor *comp*, hier nog gedeeltelijk contextueel ingevoerd, ook een volledig expliciete definitie kunnen geven:

$$compfun := [f : [x : nat]nat]\ [g : [x : nat]nat]\ [x : nat]\,\langle\langle x\rangle f\rangle g$$

Waarom beide notaties? Zoals gezegd kan λ-abstractie niet gemist worden, anders blijf je hangen in achttiende-eeuwse wiskunde. Maar de contextuele manier wil je ook behouden, die volgt getrouw de structuur van wiskundige teksten. Dit is geredeneerd vanuit het standpunt van de gebruiker. Vanuit het oogpunt van theoretische analyse van de structuur van wiskundige teksten, of van implementatie ten behoeve van machinale verificatie, ligt het anders. De Bruijn heeft dan ook veelvuldig geëxperimenteerd met formalismen waarin de beide principes geünificeerd worden. Zie ook de voorlaatste paragraaf.

Logica, dependent types

In Automath is een bewijs van een implicatie $\phi \to \psi$ een functie f die een bewijs p van ϕ overvoert in een bewijs $\langle p\rangle f$ van ψ. Dit betekent dat het type van de functies van Pr_ϕ naar Pr_ψ kan gelden als het bewijstype $Pr_{\phi\to\psi}$. In *prop*-stijl komt dit neer op identificatie van de implicatie met het functietype $[x : \phi]\,\psi$. Toepassing van de afleidingsregel modus ponens is dan gewoon functie-applicatie. Zie Figuur 4. Er is een sterke analogie tussen implicatie en universele kwantificatie. Een bewijs van $\forall x : \alpha\, P(x)$ is weer een functie f, die nu bij $a \in \alpha$ een bewijs $\langle a\rangle f$ van de propositie $P(a)$ levert. Maar er is ook een belangrijk verschil: het type van $\langle a\rangle f$, het bewijstype $Pr_{P(a)}$, zal afhangen van a. Er is geen uniform type voor de beelden van f. Bij de implicatie wel, daar is het altijd dezelfde Pr_ψ.

Voor de implicatie volstaat de typestructuur van de getypeerde λ-calculus zoals die in 1967 bestond en ook uitvoerig bestudeerd was. Voor de universele kwantificatie moest er iets nieuws komen: λ-calculus met *dependent types*. Die werd door De Bruijn bij het maken van Automath ontdekt.

Voor de formalisering in Automath moet P eerst zijn ingevoerd als propositionele functie, wat in de huidige opzet betekent: met type $[x : \alpha]$ type. De waarden van P zijn de proposities $\langle a\rangle P$, voor elke a een verschillende. Het type van een bewijs van $\forall x \in \alpha\, P(x)$ wordt nu $[x : \alpha]\,\langle x\rangle P$, een voorbeeld van een dependent type.

Net als in het geval van modus ponens komt toepassing van universele instantiatie als corresponderende afleidingsregel weer neer op functie-toepassing. Zie Figuur 5. Deze behandeling van $\to$ en $\forall$ is te zien als een directe implementatie van de functionele interpretatie in een intuïtionistische theorie van constructies. De Bruijn vermeldt, onder andere in [3], dat hij die van Heyting had geleerd en dat het een inspiratiebron was voor proofs-as-objects in Automath.

Min of meer synchroon met het ontstaan van Automath werd door logici aan die theorie van constructies gewerkt. Onafhankelijk van De Bruijn [1] werden de dependent types in een setting van propositions-as-types daarbij gevonden door Howard [6]. Howards ideeën stammen, net als die van De Bruijn, uit 1968. De correspondentie tussen proposities en types wordt vaak aangeduid als het *Curry–Howard-isomorfisme*, met soms de naam De Bruijn eraan toegevoegd.

Automath-talen en taaltheorie

In de loop der jaren is er geëxperimenteerd met diverse varianten van Automath, dat daardoor eigenlijk niet één taal is, maar een familie van verwante talen of dialecten. De verschillen zitten hem vooral in de typeringsregels en de typestructuur die daarmee verband houdt.

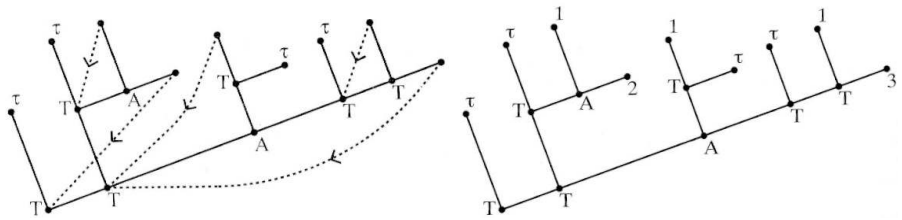
De tekstvoorbeelden in dit stuk zijn geschreven in AUT-QE, de taal die Jutting ge-

	∗	ϕ	:=	—	type
ϕ	∗	ψ	:=	—	type
ψ	∗	IMP	:=	$[x : \phi]\,\psi$	type
ψ	∗	p	:=	—	ϕ
p	∗	q	:=	—	IMP(ϕ, ψ)
p	∗	modpon	:=	$\langle p\rangle q$	ψ

Figuur 4 Modus ponens

	∗	α	:=	—	type
α	∗	P	:=	—	$[x : \alpha]$ type
P	∗	ALLE	:=	$[x : \alpha]\,\langle x\rangle P$	type
P	∗	r	:=	—	ALLE(α, P)
r	∗	a	:=	—	α
a	∗	inst	:=	$\langle a\rangle r$	$\langle a\rangle P$
	∗	*stelling2*	:=	$[x : nat]$ refl(nat, x, x)	ALLE$(nat, [x : nat]$ IS$(nat, x, x))$
	∗	*stelling1a*	:=	*inst*$(nat,$ *stelling2*$, 1)$	IS$(nat, 1, 1)$

Figuur 5 Toepassing van universele instantiatie



Figuur 6 Een term van $\Delta\Lambda$ opgeschreven als een boom

bruikte voor de Landau-vertaling. Specifiek voor AUT-QE is de wijze waarop een propositionele functie P kan worden ingevoerd met type $[x : \alpha]$ type. In het oorspronkelijke Automath, naar het jaar van ontstaan met AUT68 aangeduid, bestaat zo'n type niet.

Er is in het Automath-project uitvoerig studie gemaakt naar eigenschappen van de diverse Automath-talen zoals confluentie, normalisatie, beslisbaarheid, uniciteit van typen, enzovoorts. Onder meer de proefschriften van Rob Nederpelt en Diederik van Daalen gaan hierover. Veel is te vinden in Part C (Theory) van [8].

Ik wil hier alleen nog de aandacht vestigen op een bijzonder lid van de familie, de taal $\Delta\Lambda$, waar De Bruijn een speciale band mee had. In $\Delta\Lambda$ wordt het contextuele definitieformalisme van PAL overgenomen door de λ-notatie. Een heel Automath-boek wordt daarbij getransformeerd naar één grote λ-expressie.

In de begintijd van Automath hadden De Bruijn en Nederpelt deze mogelijkheid al onderzocht, resulterend in de verwante systemen Λ (gedefinieerd door Nederpelt) en AUT-SL. Hier staat SL voor *single line*, een boek bestaat nog maar uit één enkele regel.

In de transformatie worden context-indicatoren vervangen door λ-abstractie, waarbij de rol van instantiatie wordt overgenomen door functie-applicatie. Verder wordt, kort door de bocht, een lijn $b := P : Q$ getransformeerd naar:

- een abstractor $[b : Q]$ als b een block opener of een primitieve notion is (dus als P is — of PN)
- een applicatie/abstractor-paar (ook AT-paar genoemd) $\langle P\rangle[b : Q]$ als het een definitielijn is met P een definiërende term

Wanneer men nu bij een gedefinieerde identifier b, ergens diep in de term, de definiërende expressie P nodig heeft, dan kan dat door het AT-paar op te zoeken waar b door gebonden wordt en b door P te substitueren. Dit zou met β-reductie kunnen, maar dan worden alle andere b's met dezelfde binder ook gesubstitueerd, wat niet alleen overbodig is, maar ook ongewenst: het kan leiden tot explosieve groei van de term. Daarom is in $\Delta\Lambda$ de β-reductie vervangen door *lokale* β-reductie: de substitutie wordt lokaal uitgevoerd en het AT-paar blijft staan.

Een term van $\Delta\Lambda$ kun je opschrijven als een boom met twee soorten binaire knopen: abstractie (T) en applicatie (A). Op de eindpunten staat of type, genoteerd door τ, of een identifier, die correspondeert met een binder (T) eerder in de boom. Zie Figuur 6.

Door de boom te voorzien van interne referenties, zoals in de linker figuur door pijlen is aangegeven, kunnen de identifiers worden geëlimineerd. Die referenties kunnen ten slotte worden geïmplementeerd door zogenaamde *De Bruijn-indices*, zoals in de rechter figuur. Het getal bij een knoop geeft aan door de hoeveelste T, wandelend in de richting van de wortel van de boom, die knoop gebonden wordt.

Het resultaat is een object met een verbluffend simpele structuur: een binaire boom met vier soorten knopen: A of T op een splitsing, en het symbool τ of een getal op een eindpunt. Zo'n object kan, als single-line Automath-boek, een stuk wiskunde voorstellen. Of in de woorden van De Bruijn, in een voordracht op Heriot-Watt University, Edinburgh, november 2003: "Such a tree (with reference arrows) might represent a full mathematics encyclopaedia." De wiskunde zelf als (complexe) wiskundige structuur, opgebouwd uit een paar eenvoudige bouwstenen.

En passant is hier één van De Bruijns meest invloedrijke en meest geciteerde vernieuwingen in de λ-calculus beschreven [2], de *De Bruijn-indices*. Gebonden variabelen worden vervangen door wat De Bruijn zelf *nameless dummies* noemde, de getallen in de zojuist besproken termbomen. Hierdoor wordt ombenoeming van gebonden variabelen bij substitutie en het werken met equivalentieklassen van α-equivalente termen overbodig.

Literatuur over Automath

In het bovenstaande ben ik met opzet spaarzaam geweest met verwijzingen naar de literatuur. Er is echter veel te vinden over Automath. Met name zijn er twee zeer bruikbare bronnen.

Het boek *Selected Papers on Automath* [8] kan als een soort canonieke verslaglegging van het Automath-project worden beschouwd. De belangrijkste artikelen staan erin, alle aspecten van het project komen aan bod.

Daarnaast is er de website van het Automath Archive: www.win.tue.nl/automath. De basis van dit archief is de genummerde lijst met artikelen, rapporten en notities over Automath, die vanaf het begin werd bijgehouden. Het meeste is online te raadplegen.

Op de Radboud Universiteit Nijmegen is de groep geleid door Henk Barendregt en Herman Geuvers al jaren actief op het brede gebied van Bewijsverificatie en de daarmee gelieerde Typentheorieën. In die groep is veel kennis over Automath aanwezig. Freek Wiedijk schreef niet alleen een nieuwe checker voor AUT68 en AUT-QE. Zijn website www.cs.ru.nl/~freek bevat ook een schat aan informatie over proof checkers, met inbegrip van Automath.

Referenties

1 N.G. de Bruijn, The mathematical language AUTOMATH, its usage, and some of its extensions, in M. Laudet, ed., *Proceedings of the Symposium on Automatic Demonstration, held December 1968, Versailles, France*, Springer-Verlag LNM 125, 1970, pp. 29–61.

2 N.G. de Bruijn, Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem, *Indag. Math.*, 34(5), 1972, 381–392.

3 N.G. de Bruijn, Gedachten rondom AUTOMATH, Rapport V40, TU Eindhoven, Faculteit Wiskunde en Informatica, 1990.

4 N.G. de Bruijn, A plea for weaker frameworks, In G. Huet and G. Plotkin, eds., *Logical Frameworks*, Cambridge University Press, 1991, pp. 40–67.

5 N.G. de Bruijn, Type-theoretical checking and philosophy of mathematics, In G. Sambin and J. Smith, eds, *Twenty-Five Years of Constructive Type Theory*, Oxford University Press, August 1998, Proceedings of a Congress held in Venice, Italy, October 1995, pp. 41–56.

6 W. Howard, The formulae-as-types notion of construction, in R. Hindley and J. Seldin, eds., *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, 1980, pp. 107–124.

7 E. Landau, *Grundlagen der Analysis*, Chelsea Publ. Comp., 3rd edition, 1960.

8 R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, eds. *Selected Papers on Automath*, Vol. 133 of *Studies in Logic and the Foundations of Mathematics* Elsevier, 1994.

9 Freek Wiedijk, A new implementation of automath, *J. Autom. Reasoning*, 29(3-4), 2002, 365–387.