

Definition of limits in Automath

Citation for published version (APA):

de Bruijn, N. G. (1970). *Definition of limits in Automath*. Technische Hogeschool Eindhoven.

Document status and date:

Published: 01/01/1970

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Definition of LIMITS in AUTOMATH

AUTOMATH

by N.G. de Bruijn. *

1. Introduction to mathematics.

1.1	O	bool	:=	PN	<u>sort</u>
1.2	O	x	:=	- - - - -	bool
1.3	x	TRUE	:=	PN	<u>sort</u>
1.4	O	ksi	:=	- - - - -	<u>sort</u>
1.5	ksi	nonempty	:=	PN	bool
1.6	ksi	a	:=	- - - - -	ksi
1.7	a	then 2	:=	PN	TRUE(nonempty)
1.8	ksi	P	:=	- - - - -	[u ksi] bool
1.9	P	exists	:=	PN	bool
1.10	P	v	:=	- - - - -	ksi
1.11	v	ass 1	:=	- - - - -	TRUE({v} P)
1.12	ass 1	then 13*	:=	PN	TRUE(exists)
1.13	P	ALL	:=	[u ksi] TRUE({u} P)	<u>sort</u>
1.14	P	all	:=	nonempty(ALL)	bool
1.15	O	a	:=	- - - - -	bool
1.16	e	b	:=	- - - - -	bool
1.17	b	IMPL	:=	[u TRUE(a)] TRUE(b)	<u>sort</u>
1.18	b	impl	:=	nonempty(IMPL)	bool

* Internal Report, 1969
 Department of Mathematics,
 Technological University
 Eindhoven, The Netherlands.

2. Definition of limit.

2.1	0	real	:=	PN	<u>sort</u>
2.2	0	r1	:=	- - - - -	real
2.3	r1	r2	:=	- - - - -	real
2.4	r2	distance	:=	PN	real
2.5	r2	less	:=	PN	bool
2.6	0	null	:=	PN	real
2.7	0	nat	:=	PN	<u>sort</u>
2.8	0	sequence	:=	[n nat]real	<u>sort</u>
2.9	0	1	:=	PN	nat
2.10	0	k	:=	- - - - -	nat
2.11	k	l	:=	- - - - -	nat
2.12	1	lessnat	:=	PN	bool
2.13	0	a	:=	- - - - -	sequence
2.14	a	l	:=	- - - - -	real
2.15	l	eps	:=	- - - - -	real
2.16	eps	m ₀	:=	- - - - -	nat
2.17	m ₀	m	:=	- - - - -	nat
2.18	m	w	:=	impl(lessnat(m ₀ ,m), less(distance({m}a,l),eps))	bool
2.19	m ₀	z	:=	all(nat,[s nat] w(s))	bool
2.20	eps	y	:=	exists(nat,[t nat] z(t))	bool
2.21	eps	q	:=	impl(less(null,eps),y)	bool
2.22	l	lim	:=	all(real,[eps,real] q(eps))	bool
2.23	a	conv	:=	exists(real,[t real] lim(t))	bool

3. A lemma without proof.

3.1	0	x	:=	-----	real
3.2	x	y	:=	-----	real
3.3	y	if	:=	-----	TRUE(less(null,y))
3.4	if	lemma	:=	PN	TRUE(less(distance(x,x),y))

4. Constant sequences are convergent.

4.1	0	c	:=	-----	real
4.2	c	p	:=	[m nat] c	sequence
4.3	c	delta	:=	-----	real
4.4	delta	ass	:=	-----	TRUE(less(null,delta))
4.5	ass	n ₀	:=	-----	nat
4.6	n ₀	n	:=	-----	nat
4.7	n	then	:=	lemma({n}p,delta,ass)	TRUE(less(distance ({n}p,c),delta))
4.8	n	a	:=	lessnat(n ₀ ,n)	bool
4.9	n	aa	:=	w(p,c,delta,n ₀ ,n)	bool
4.10	n	b	:=	then 2([u TRUE(a)] <u>sort</u> (then), [u TRUE(a)] then)	TRUE(aa)
4.11	n ₀	h	:=	then 2([s nat] ^{TRUE(c)} <u>aa</u> (s),[s nat]b(s))	TRUE(z(p,c,delta,n ₀))
4.12	ass	d	:=	then 13*(nat,[x nat]z(p,c,delta,1),1,h(1))	TRUE(y(p,c,delta))
4.13	delta	e	:=	then 2([c <u>sort</u> (ass)] ^(c) <u>sort</u> (d), [c <u>sort</u> (ass)]d(c))	TRUE(q(p,c,delta))
4.14	c	f	:=	then 2([s real] TRUE(q,(p,c,s)), [s real] e(s))	TRUE(lim(p,c))
4.15	c	g	:=	then 13*(real,[s real] lim(p,s),c,f)	TRUE(conv(p))

Remark. If with a given indicator a line $a := \Sigma \wedge$ is acceptable, then we take the liberty to write sort(Σ) instead of \wedge . Example: (see b above) : sort(then) stands for TRUE(less(distance({n}p,c),delta)).