Programming primitive recursive functions and beyond

Hannes Hutzelmeyer

Summary

In addition to the publication ***The Snark, a counterexample for Church's thesis ?*** examples and details are offered in the form of two appendices C6 and C7 that allow for better understanding of the general method and the particular problem related to Church's thesis.

The author has developed an approach to logics that comprises, but also goes beyond predicate logic. The **FUME** method contains two tiers of precise languages: object-language **Funcish** and metalanguage **Mencish**. It allows for a very wide application in mathematics from recursion theory and axiomatic set theory with first-order logic, to higher-order logic theory of real numbers etc.

The most usual approach to calculative (effectively calculable) functions is done by register machines or similar storage-based computers like the Abacus or Turing machines. Another usual approach to computable functions is to **start** with **primitive recursive functions**. However, one has to find a way to put this into a form that does not rely on a pre-knowledge about functions and higher logic. The concrete calcule LAMBDA of decimal pinitive arithmetic allows for such an access. It is based on a machine that is completely different from the storage-based machines: the PINITOR does not use storages but rather many microprocessors, one for each appearance of a command in the code of primitive recursive functior. the codes are decimal numbers, called **pinons** , where only the characters `0 1 2 8 9` appear. There are four kind of commands only: `0` nullification, `1` succession, `2` straight recursion and `8` composition. The PINITOR is a **calculator** which means that there is no halting problem. Computers have halting problems, per defintion calculators do not.

Appendix C6 gives the **programming** of the codes of most of the usual primitive function and goes even farther, e.g. it introduces **generator** technique that allows for the straight-forward calculation of so-called **processive** function, that are not primitive recursive. The most famous examples of Ackermann and other hyperexponential functions are programmed.

Appendix C7 turns to the **Boojum-function** and the **Snark-function** that have been introduced as calculative functions in the above publication in connection with Church's thesis. A list is provided that gives the lowest values for these functions and gives some more insight into these functions.

Contact: Hutzelmeyer@pai.de
https://pai.de

C6 Programming pinon strings

A ***number*** string can be referred to in Funcish and Mencish, either by an ***individual-constant*** e.g. Λbpt or a macro, e.g. ***Λbpt*** resp. . The Mencish notation with macros is shorter, as one can include ***individual-constant*** strings in Funcish only by means of synaption, that is denoted by (Λ∗Λ) e.g.

Λbpt ***=*** 2***ΛufcΛbpc*** ***=*** 2{10}20{11} *notice font style boldface italics for the equaliser in Mencish* ***=***
Λbpt=((2∗Λufc)∗Λbpc)=2{10}20{11} *different from normal font for equaliser in Funcish =*

The first examples show the usual claim that recursive functions need **constant functions** (otherwise called constantions or better fications) and **projection functions** to start with is **wrong**. It suffices to have **nullification** and **succession**. Fications and projection are obtained therefrom by primitive recursion, coded by ***pinon*** strings. E.g. identitation *f(x)=x* is programmed by 201 , constant *1* by 8109

Mencish	*Funcish*	*primitive recursive function* *conventional notation*	***pinon*** *codes as Mencish strings*	*intr. arity*
Λn	Λn Λnfc	nullification, constantion 0	0	0
Λufc	Λufc	unification, constantion 1	{10} = 8109 [1)]	0
Λbfc	Λbfc	duofication, constantion 2	{1{10}}	0
Λtfc	Λtfc	trification, constantion 3	{1{1{10}}}	0
Λqfc	Λqfc	quadrufication, constantion 4	{1{1{1{10}}}}	0
		...		
Λouufc	Λouufc	811-fication, constantion 811	{1 ... *811 times* ... 0 ... *811 times*... }	0
Λu	Λu Λucs	succession, unicession x+1	1	1
Λbcs	Λbcs	bicession x+2	{11}	1
Λtcs	Λtcs	tricession x+3	{1{11}}	1
		...		
Λupr	Λupr	identation, uni-projection x	201	1
Λbpr	Λbpr	bi-projection y	2201201	2
Λtpr	Λtpr	tri-projection z	22201201201	3
Λqpr	Λqpr	quadru-projection z	222201201201201	4
		...		
Λbpc	Λbpc	duplication 2x	20{11}	1
Λtpc	Λtpc	triplication 3x	20{1{11}}	1
		...		
Λcbp	Λcbp	cession-duplication 2x+1	{120{11}}	1

1) for better readability synonymous usage of { } and 8 9 is employed for composition

Table C6.1.1 Constantions and strictly ascending arithmetic functions (to be continued)

Λsad	Λsad	succession-addition x+y+1	211	2
Λadd	Λadd	addition x+y	22011	2
Λouufca	Λouufca	alternative 811-fication	{***Λadd***{***Λopc***{***ΛdpcΛdfc***}}{***ΛaddΛdfcΛufc***}}	0
Λtmad	Λtmad	ternary-addition x+y+z	2220111	3
Λqmad	Λqmad	quaternary-addition x+y+z+w	222201111	4
Λpmad	Λpmad	quintary-addition x+y+z+v+w	22222011111	5
		...		
Λtpad	Λtpad	ternary-pair-addition x+z	222012011	3
Λqpad	Λqpad	quaternary-pair-addition x+w	2222012012011	4
		...		
Λmula	Λmula	multiplication x.y alternative	20{***ΛaddΛuprΛtpr***}	2
Λmul	Λmul	multiplication xy	20***Λtpad*** = 20222012011	2
Λsup	Λsup	supplication (x+1)y	2201***Λtpad*** = 2201222012011	2
Λbpo	Λbpo	dual power,squaring,quadration x^2	{***ΛmulΛuprΛupr***}	1
Λtpo	Λtpo	tertial power,cubing, cubation x^3	{***ΛmulΛuprΛbpo***}	1
Λqpo	Λqpo	quartal power (potention) x^4	{***ΛmulΛuprΛtpo***}	1
		...		
Λdpo	Λdpo	decimal power (potention) x^{10}	{***ΛmulΛuprΛvpo***}	1
		...		
Λbpt	Λbpt	bi-ponentiation 2^x	2{10}***Λbpc*** =2{10}20{11}	1
Λtpt	Λtpt	tri-ponentiation 3^x	2{10}***Λtpc*** =2{10}20{1{11}}	1
Λdpt	Λdpt	deci-ponentiation 10^x	2{10}***Λdpc***	1
		...		
Λsbpt	Λsbpt	super-bi-ponentiation 2^^x	2{10}***Λbpt*** = 2{10}2{10}20{11}	1
Λssbpt	Λssbpt	supersuper-bi-ponentiation 2^^^x	2{10}***Λsbpt*** = 2{10}2{10}2{10}20{11}	1
		...		
Λcarl	Λcarl	carlation[1)] (x(x+1))/2	20***Λsad***	1
Λincarl	Λincarl	incarlation (x(x+1))/2+y+1	21***Λsad*** = 21211	2
Λpcarl	Λpcarl	predecessor carlation (x(x-1))/2	20***Λadd***	1
Λfact	Λfact	factorial, factorialation x!	2{10}{***ΛmulΛupr***{1***Λbpr***}}	1
Λjexp	Λjexp	trans-exponentiation[2)] y^x=y^x	2{10}{***ΛmulΛuprΛtpr***}	2
Λexp	Λexp	exponentiation x^y=x^y	{***ΛjexpΛbprΛupr*** }	2
Λautxp	Λautxp	auto-ponentiation x^x	{***ΛjexpΛupr Λupr*** }	1
Λbla	Λbla	dual ladder $2_x(y)$, escalation	2***ΛuprΛbpt***	2
Λtla	Λtla	tertial ladder $3_x(y)$	2***ΛuprΛtpt***	2
Λqla	Λqla	quartal ladder $4_x(y)$	2***ΛuprΛqpt***	2
		...		
Λjsexp	Λjsexp	trans-super-exponentiation y^^x	2{10}{***ΛjexpΛuprΛtpr*** }	2
Λjssexp	Λjssexp	trans-supersuper-exponentiation y^^^x	2{10}{***ΛjsexpΛuprΛtpr*** }	2
Λjsssexp	Λjsssexp	trans-supersupersuper-exponentiation y^^^^x	2{10}{***ΛjssexpΛuprΛtpr*** }	2
		...		
Λsexp	Λsexp	super-exponentiation x^^y	{***ΛjsexpΛbprΛupr***}	2
Λssexp	Λssexp	supersuper-exponentiation x^^^y	{***ΛjssexpΛbprΛupr***}	2
Λsssexp	Λsssexp	supersupersuper-exponentiation x^^^^y	{***ΛjsssexpΛbprΛupr***}	2

[1)] *remember little Carl Gauss* [2)] *transposed exponentiation*

Table C6.1.1 Constantions and strictly ascending arithmetic functions (continuation)

Λngy	Λngy	negation characteristic 1,0,0,... truncated [1] <1-x>	2{10}0	1
Λsgy	Λsgy	signation characteristic 0,1,1,... <1-<1-x>>	20{10}	1
Λcjy	Λcjy	conjunction characteristic	{***ΛsgyΛadd*** }	2
Λdjy	Λdjy	disjunction characteristic	{***ΛsgyΛmul***}	2
Λipy	Λipy	implication characteristic	{***ΛdjyΛngyΛbpr***}	2
Λbcy	Λbcy	bicondition characteristic	{***ΛcjyΛipy***{***ΛipyΛbprΛbur***}}	2
Λudc	Λudc	uni-decession, predec. <x-1> 0,0,1,2,...	20***Λbpr***	1
Λbdc	Λbdc	bi-decession <x-2> 0,0,0,1,2,...	{***ΛudcΛudc***}	1
Λtdc	Λtdc	tri-decession <x-3> 0,0,0,0,1,2,...	{***ΛudcΛbdc***}	1
Λjsub	Λjsub	transposed truncated subtraction [2] <y-x>	2***ΛuprΛudc***	2
Λsub	Λsub	truncated subtraction [3] <x-y>	{***ΛjsubΛbprΛupr***}	2
Λadi	Λadi	absolute differencition <y-x>+<x-y>	{***ΛaddΛsubΛjsub***}	2
Λemax	Λemax	equi-maximation of two numbers	{***ΛaddΛsubΛbpr***}	2
Λemin	Λemin	equi-minimation of two numbers	{***ΛsubΛaddΛemax***}	2
Λeqy	Λeqy	equality characteristic x=y	{***ΛsgyΛadi***}	2
Λieqy	Λieqy	inequality characteristic x≠y	{***ΛngyΛadi***}	2
Λmiy	Λmiy	minority characteristic x<y	{***ΛngyΛjsub***}	2
Λemiy	Λemiy	equal-minority characteristic x=<y	{***ΛsgyΛsub***}	2
Λmay	Λmay	majority characteristic y<x	{***ΛngyΛsub***}	2
Λemay	Λemay	equal-majority charact. y=<x	{***ΛsgyΛjsub***}	2
Λtangy	Λtangyy	triangularity	{***Λcjy***{***ΛmiyΛtpr***{***ΛaddΛbprΛtpr***}} {***ΛmiyΛbpr***{***ΛaddΛtprΛupr***}}}	3
Λpythy	Λpythy	Pythagoras triple	{***ΛeqyΛbpo*** {***Λadd***{***ΛbpoΛbpr***}{***ΛbpoΛtpr***}}}	3
Λugy	Λugy	inequality unus characteristic, not =1	{***ΛaddΛngy***{***ΛsgyΛudc***}}	1
Λbgy	Λbgy	inequality duo characteristic, not =2	{***ΛugyΛudc***}	1
Λtgy	Λtgy	inequality tres characteristic, not =3	{***ΛbgyΛudc***}	1
		...		
Λuqy	Λuqy	equality unus characteristic =1	{***ΛngyΛugy***}	1
Λbqy	Λbqy	equality duo characteristic =2	{***ΛngyΛbgy***}	1
		...		
Λumay	Λumay	majority unus 1<x 1 1 0 0 0 …	{***ΛngyΛudc***}	1
Λbmay	Λbmay	majority duo 2<x 1 1 1 0 0 …	{***ΛngyΛbdc***}	1
		...		
Λody	Λody	1 0 1 0 1 ... oddity characteristic	2{10}***Λngy***	1
Λevy	Λevy	0 1 0 1 0 ... evenness characteristic	20***Λngy***	1
Λbmp	Λbmp	2 0 1 2 3 ... <x-1>+<1-x>)	{***ΛaddΛudc***{***ΛbpcΛngy***}}	1
Λtmp	Λtmp	3 0 1 2 3 ... <x-1>+3<1-x>	{***ΛaddΛudc***{***ΛtpcΛngy***}}	1
Λtrp	Λtrp	2 1 0 2 1 0 ...	2***Λdfc***{***ΛbmpΛupr***}	1
Λtxy	Λtxy	0 1 0 0 1...	{***Λngy*** {***Λtrp***1}}	1
Λqxy	Λqxy	0 0 1 0 0 0 1...	{***Λngy***{***Λqrp***1}}	1
		[1] *angle brackets < >denote truncation*	[2] *short: transtraction* [3] *short: subtraction*	

Table C6.1.2 Subtractive and junctive logic algebra arithmetic functions

		number-constant		
Λzlinv	Λzlinv	left auxiliary entire [1] inverse	20{***ΛaddΛupr***{***Λngy***{***Ajsub***{1***Λbpr***}{	
Λzruinv	Λzruinv	right auxiliary unary entire inverse	1}}}}	
Λzrbinv	Λzrbinv	right auxiliary binary entire inverse	1***Λtpr***}}}}	
		unary entire inversion of ***Λ1***	***Λzlinv Λ1 Λzruinv***	1
		binary entire inversion of ***Λ1***	***Λzlinv Λ1 Λzrbinv***	2
Λbsc	Λbsc	entire bi-section [x/2] [2]	***Λzlinv Λbfc Λzruinv***	1
Λtsc	Λtsc	entire tri-section [x/3]	***Λzlinv Λtfc Λzruinv***	1
Λdsc	Λdsc	entire deci-section [x/10]	***Λzlinv Λdfc Λzruinv***	1
		...		
Λwcarl	Λwcarl	inverse carlation	***Λzlinv Λcarl Λzruinv***	1
Λwfact	Λwfact	inverse factorialation	***Λzlinv Λfact Λzruinv***	1
Λbrt	Λbrt	entire bi-radication [^{2}rt(x)]	***Λzlinv Λbpo Λzruinv***	1
Λtrt	Λbrt	entire tri-radication [^{3}rt(x)]	***Λzlinv Λtpo Λzruinv***	1
Λqrt	Λqrt	entire quadri-radication [^{4}rt(x)]	***Λzlinv Λqpo Λzruinv***	1
		...	...	
Λblr	Λblr	entire bi-logarithmation [$\log_2$x]	***Λzlinv Λbpt Λzruinv***	1
Λtlr	Λtlr	entire tri-logarithmation [$\log_3$x]	***Λzlinv Λtpt Λzruinv***	1
Λdlr	Λdlr	entire deci-logarithmat.[$\log_{10}$x]	***Λzlinv Λdpt Λzruinv***	1
		...	...	
Λtscr	Λtscr	entire tri-section remainder	{***ΛsubΛupr***{***ΛtfcΛtsc***}}	1
Λtrtr	Λtrtr	entire tri-radication remainder	{***ΛsubΛupr***{***ΛtpoΛtrt***}}	1
Λtlrr	Λtlrr	entire tri-lorgaithmat. remainder	{***ΛsubΛupr***{***ΛtptΛtlr***}}	1
		...		
Λtscy	Λtscy	entire tri-sectibility	{***ΛsgyΛtscr***}	1
Λtrty	Λtrty	entire tri-rootability	{***ΛsgyΛtrtr***}	1
Λtlry	Λtrty	entire tri-lorithmability	{***ΛsgyΛtlrr***}	1
		other prefixes accordingly		1
Λdiv	Λdiv	entire division [x/y] x for y=0	***Λzlinv Λmul Λzrbinv***	2
Λjdiv	Λjdiv	transposed entire division [y/x]	{***ΛdivΛbprΛupr***}	2
Λdir	Λdir	divison remainder x-y[x/y]	{***ΛsubΛupr***{***ΛmulΛbprΛdiv***}}	2
Λdivy	Λdivy	divisibility of x by y no for y=0 x=1…, yes for [0/0]	{***ΛsgyΛdir***}	2
Λidivy	Λidivy	indivisibility of x by y	{***ΛngyΛdir***}	2
Λmodcgy	Λmodcgy	modulo-congruity x=y mod z	{***ΛdivyΛadiΛtpr***}	1
Λrad	Λrad	entire radication [^{y}rt(x)] , x for y=0	***Λzlinv Λexp Λzrbinv***	2
Λlog	Λlog	entire logarithmation [$\log_y$x] , x for y=0	***Λzlinv Λjexp Λzrbinv***	2
Λjrad	Λjrad	transp.e.radication [^{x}rt(y)]	{***ΛradΛbprΛupr***}	2
Λjlog	Λjlog	transp.e.logarithmation [$\log_x$y]	{***ΛlogΛbprΛupr***}	2
		[1] *short: the word 'entire' is left awa y*	[2] *square brackets denote entire part*	

Table C6.2.1 Entire inversion of strictly ascending functions

The following table makes use of such inversions:

		synaption		
Λbsa	Λbsa	dual synaption	{***Λadd***{***ΛmulΛupr***{***Λbpt***{1{***ΛblrΛbpr***}}}}***Λupr***}	2
Λosa	Λosa	octal synaption	{***Λadd***{***ΛmulΛupr***{***Λopt***{1{***ΛolrΛbpr***}}}}***Λupr***}	2
Λdsa	Λdsa	decimal synaption e.g. Λdsa(210;90)=21090 alternatively (210∗90)=21090	{***Λadd***{***ΛmulΛupr***{***Λdpt***{1{***ΛdlrΛbpr***}}}}***Λupr***}	2
Λbdip	Λbdip	dual digit of x at position[1)] y dual suite decode bdip(x,y) code x, position y, length $\log_2(x)+1$ e.g. 1 1 0 0 1 0 1 0 interpreted as dual number gives decimal number code 210; it has length 8 and e.g. digit 1 at position 3	{***Λdiv***{***ΛdirΛupr***{***Λbpt***{1***Λbpr***}}}{***ΛbptΛbpr***}}	2
Λddip	Λddip	decimal digit of x at position y	{***Λdiv***{***ΛdirΛupr***{***Λdpt***{1***Λbpr***}}}{***ΛdptΛbpr***}}	2
		tuple-pair coding		
Λadpair	Λadpair	antidiagonal-pair code pair(j,k) = j+((j+k)(j+k+1))/2 Cantor pairing function	{***ΛaddΛupr***{***ΛcarlΛadd***}}	2
Λxadrt	Λxadrt	antidiagonal auxiliary root r(n) = [(²rt(8n+1)-1)/2]	{***Λbsc***{***Λudc***{***Λbrt***{1***Λofc***}}}}	1
Λadrow	Λadrow	row antidiagonal method [n-(r(n)(r(n)+1))/2]	{***ΛsubΛupr***{***ΛcarlΛxadrt***}}	1
Λadcol	Λadcol	column antidiagonal method [((r(n)+1)(r(n)+2))/2-(n+1)]	{***Λsub***{***Λcarl***{1***Λxadrt*** }}1}	1
Λadt	Λadt	triple p.coding pair(pair(j,k),l)	{***Λadpair Λadpair Λtpr***}	3
Λadq	Λadq	quadruple pair coding …	{***Λadpair Λadt Λqpr***}	4
Λfibo	Λfibo	Fibonacci sequence 1,1,2,3,5,8,… f(0)=1 f(1)=1 f(i+2)=f(i)+f(i+1)	{***Λadrow*** 2{1{1{1{10}}}} {***Λadpair***{***Λadcol***{***Λadd Λadrow Λadcol***}}}}	1
		1) 'position' starts at 0, 'place' at 1		

Table C6.2.2 Synaption and tuple-pair coding

The following table C62.3 shows how pinity ***Λpiny*** is programmed, that allows to replace #Λ . All necessary ***pinon*** strings have been defined above (top-down principle). With synaptive recursion of Mencish the definition is very simple:

pinon :: 0 ¦ 1 ¦ 2 ***pinon pinon*** ¦ { ***pinon pinon-desmos*** }
pinon-desmos :: ***pinon*** ¦ ***pinon-desmos pinon***

This has to be expressed by a primitive recursive characteristic function Λpiny(Λ). Λpiny is defined in the following table (not top-down within the table).

Λpiny	pinity characteristic [#Λ1]↔[Λpiny(Λ1)=0]	{***Λuqy***{2***Λxnu-repl***{***Λxouuu-repl*** {***Λxouuv-replΛxbuu-repl***}}***ΛdlrΛupr***}}	2
	the programming for the four necessary auxiliary ***pinon*** strings is shown below		
	four full replacements		
Λxnu-repl	replace all characters 0 by 1 , deci-lorithmation is used for the limit of recursion	{2201{***Λxnu-preplΛupr***{***ΛudcΛbpr***}}{***1Λdlr***}***Λupr***}	1
Λxbuu-repl	replace from right 211 by 1	{2201***Λxbuu-preplΛdlrΛup***r}}	1
Λxouuv-repl	replace from right {11} by 1	{2201***Λxouuv-preplΛdlrΛupr***}}	1
Λxouuu-repl	replace from right {111 by {11	{2201***Λxouuu-preplΛdlrΛup***r}}	1
	auxiliaries for the four replacements		
Λxbuu-eqy	characteristic equality 211	{***ΛadiΛupr***{***ΛdsaΛbpc***{***ΛdsaΛufcΛufc***}}}	1
Λxouuv-eqy	characteristic equality {11}	{***ΛadiΛupr***{***ΛdsaΛopc***{***ΛdsaΛufc***{***ΛdsaΛufcΛofc***}}}}	1
Λxouuu-eqy	characteristic equality {111	{***ΛadiΛupr***{***ΛdsaΛopc***{***ΛdsaΛufc***{***ΛdsaΛufcΛufc***}}}}	1
Λxbuu-u	function that maps 201 to 1 , others to themselves	{***Λadd***{***ΛmulΛxbuu-uqyΛupr***}{***ΛngyΛxbuu-eqy***}}	1
Λxouuv-u	function that maps {11} to 1 , others to themselves	{***Λadd{ΛmulΛxouuv-eqyΛupr***} {***ΛngyΛxouuv-eqy***}}	1
Λxouuu-ouu	function that maps {111 to {11 others to themselves	{***Λadd***{***ΛmulΛxouuu-eqyΛupr***}{***ΛmulΛouufca*** {***ΛngyΛxouuu-eqy***}}	1
	four position replacements		
Λxnu-prepl	replace 0 by 1 in digit-position Λ1 of Λ2[1)], no change otherwise	{***ΛaddΛbpr***{***ΛmulΛdpt***{***ΛemiyΛdpt***{***ΛsubΛbpr*** {***Λmul***{***Λdpt***1}{***ΛdivΛbpr***{***Λdpt***1}}}}}}}	2
Λxbuu-prepl	replaces 211 by 1 left of digit-position Λ1 of Λ2 , no change otherwise [2)]	{***Λdsc***{***Λdsa***{***ΛdivΛbpr***{***ΛdptΛqcs***}}{***Λdsa*** {***Λxbuu-u***{***Λdiv***{***ΛsubΛbpr***{***Λmul***{***ΛdivΛbpr*** {***ΛdptΛqcs***}}{***ΛdptΛqcs***}}}{***Λdpt***1}}}{***ΛsubΛbpr*** {***Λmul***{***ΛdivΛbpr***{***Λdpt***1}}{***Λdpt***1}}}}}***Λupr*** {***ΛdpcΛbpr***}}	2
Λxouuv-prepl	replaces {11} by 1 left of digit-position Λ1 of Λ2 , no change otherwise [2)]	{***Λdsc***{***Λdsa***{***ΛdivΛbpr***{***ΛdptΛpcs***}}{***Λdsa*** {***Λxouuv-u***{***Λdiv***{***ΛsubΛbpr***{***Λmul***{***ΛdivΛbpr*** {***ΛdptΛpcs***}}{***ΛdptΛpcs***}}}{***Λdpt***1}}}{***ΛsubΛbpr*** {***Λmul***{***ΛdivΛbpr***{***Λdpt***1}}{***Λdpt***1}}}}}***Λupr*** {***ΛdpcΛbpr***}}	2
Λxouuu-prepl	replaces {111 by {11 left of digit-position Λ1 of Λ2 , no change otherwise [2)]	{***Λdsc***{***Λdsa***{***ΛdivΛbpr***{***ΛdptΛpcs***}}{***Λdsa*** {***Λxouuu-ouu***{***Λdiv***{***ΛsubΛbpr***{***Λmul***{***ΛdivΛbpr*** {***ΛdptΛpcs***}}{***ΛdptΛpcs***}}}{***Λdpt***1}}}{***ΛsubΛbpr*** {***Λmul***{***ΛdivΛbpr***{***Λdpt***1}}{***Λdpt***1}}}}}***Λupr*** {***ΛdpcΛbpr***}}	2

[1)] digit-position from right [2)] for avoiding synaption problem 0 is attached at start to the right and at the end removed

Table C6.2.3 Pinity as an example for synaptic recursion (of pinon *strings)*

		auxiliary	***number-constant***
Λzllisu	Λzllisu	left limited sum	{20{***ΛaddΛupr***{
Λzllipr	Λzllipr	left limited product	{2{10}{***ΛmulΛupr***{
Λzrulisp	Λzrulisp	right unary limited sum, product	***Λbpr***}}1}
Λzrblisp	Λzrblisp	right binary limited sum, product	***ΛbprΛtpr***}}1***Λbpr***}
Λzrtlisp	Λzrtlisp	right ternary limited sum, product	***ΛbprΛtprΛqpr***}}1***ΛbprΛtpr***}
Λzcuflisp	Λzcuflisp	center unary function-lim. sum, product	***ΛbprΛtpr***}}{1
Λzruflisp	Λzruflisp	right unary function-lim. sum, product	}***Λupr***}
Λzrualisp	Λzruflisp	right unary argument-lim. sum, product	***ΛbprΛtpr***}}1***Λupr***}
Λzcbflisp	Λzcbflisp	center binary function-lim. sum, product	***ΛbprΛtprΛqpr*** }}{1
Λzrbflisp	Λzrbflisp	right binary function-lim. sum, product	}***ΛuprΛbpr***}
Λzlliom	Λzliom	left limited omnitive	20{***ΛcjyΛupr***{
Λzllien	Λzlien	left limited entitive	2{10}{***ΛdjyΛupr***{
Λzcbliqu	Λzcbliqu	center binary limited quantive	***Λbpr***}}
Λzctliqu	Λzctliqu	center ternary limited quantive	***ΛbprΛtpr***}}
Λzcqliqu	Λzcqliqu	center quatermary limited quantive	***ΛbprΛtprΛqpr***}}
Λzllimi	Λzllimi	left limited minimization	***ΛzllisuΛzllien***
Λzrtlimi	Λzrtlimi	right ternary limited minimization	***Λzrtlisp Λzrtlisp***
Λzrblimi	Λzrblimi	right binary limited minimization	***Λzrblisp Λzrblisp***
Λzctflimi	Λzctflimi	center ternary function-limit-minimization	***Λzrtlisp ΛbprΛtprΛqpr*** }}{1
Λzrtflimi	Λzrtflimi	right ternary function-limited minimization	}***ΛbprΛtpr*** }
Λzruvlimi	Λzruvlimi	right unary variable-limited minimization	***Λzrblisp Λzrualisp***
Λzcuflimi	Λzcuflimi	center unary function-limited minimization	***Λzrblisp Λzcuflisp***
Λzrbvlimi	Λzrbvlimi	right binary variable-limited minimization	***Λzrtlisp ΛbprΛtprΛqpr*** }}1***ΛuprΛbpr*** }
Λzllima	Λzllima	left limited maximization	{***ΛsubΛupr Λzllimi*** {
Λzrbvlima	Λzrbvlima	rigth binary variable-limited maximization	{***ΛsubΛbprΛupr***}***Λtpr***}***Λzrbvlimi*** }
Λzldenu	Λzldenu	left denumeration	20***Λzllimi***{***Λcjy***
Λzcdenu	Λzcdenu	center denumeration	***Λmay***}***Λzcuflimi***

limited sum or product	***pinon***	
binary function f(x,y) by limited sum of h(i,y) given by ***pinon Λ1***, i from 0 up to x[1)]	***Λzllisu Λ1 Λzrblisp***	2
unary function f(x) by function-limited sum of h(i,x) given by ***pinon Λ1***, i from 0 to g(x) [1)] given by ***pinon Λ2***	***Λzllisu Λ1 Λzcuflisp Λ2 Λzruflisp***	1
ternary function f(x,y,z) by limited sum of h(i,y,z) , i from 0 up to x	***Λzllisu Λ1 Λzrtlisp***	3
binary function f(x,y) by function-limited sum of h(i,x,y) , i from 0 to g(x)	***Λzllisu Λ1 Λzcbflisp Λ2 Λzrbflisp***	2
binary function f(x,y) by limited product of h(i,y) i from 0 up to x	***Λzllipr Λ1 Λzrblisp***	2
unary function f(x) by function-limited product of h(i,x) , i from 0 to g(x)	***Λzllipr Λ1 Λzcuflisp Λ2 Λzruflisp***	1

other limited sums and products of higher arity analogously

[1)] *including the limits*

Table C6.2.4 Programming with limits (to be continued)

limited-quantive-phrase *strings*	***pinon***	
unary characteristic function f(x) replacing omnitive case with a unary function h(i) given by ***Λ1*** $\forall\Lambda_2[[\Lambda_2<\Lambda_1]\rightarrow[\Lambda_1(\Lambda_2)=0]]$	***Λzlliom Λ1 Λbpr***}}	1
binary function h(i,j) by ***Λ1*** $\forall\Lambda_2[[\Lambda_2<\Lambda_1]\rightarrow[\Lambda_1(\Lambda_2;\Lambda_3)=0]]$	***Λzlliom Λ1 ΛbprΛtpr***}}	2
unary characteristic function f(x) replacing entitive case with a unary function h(x) given by ***Λ1*** $\exists\Lambda_2[[\Lambda_2<1(\Lambda_1)]\wedge[\Lambda_1(\Lambda_2)=0]]$	***Λzllien Λ1 Λbpr***}}	1
fication ***pinon Λ4*** for ***number Λ2*** $\forall\Lambda_1[[\Lambda_1<\Lambda_2]\rightarrow[\Lambda_1(\Lambda_1)=0]]$	{***Λzlliom Λ1 Λbpr***}} ***Λ4***}	0
$\forall\Lambda_2[[\Lambda_2<\Lambda_2(\Lambda_1)]\rightarrow[\Lambda_1(\Lambda_2)=0]]]$ ***pinon Λ2***	{***Λzlliom Λ1 Λzculiqu Λ2*** }	1
$\forall\Lambda_3[[\Lambda_3<\Lambda_2(\Lambda_1)]\rightarrow[\Lambda_1(\Lambda_3;\Lambda_2)=0]]$	{***Λzlliom Λ1 Λzcbliqu Λ2*** }	2
$\forall\Lambda_2[[\Lambda_2<\Lambda_2(\Lambda_1)]\rightarrow[\Lambda_1(\Lambda_2;\Lambda_1)=0]]$	{{***Λzlliom Λ1 Λzctliqu Λ2*** }***ΛuprΛupr***}	1
$\forall\Lambda_3[[\Lambda_3<\Lambda_2(\Lambda_1)]\rightarrow[\Lambda_1(\Lambda_3;\Lambda_1;\Lambda_2)=0]]$	{{***Λzlliom Λ1 Λzcqliqu Λ2*** }***ΛuprΛuprΛbpr***}	2
higher arities analogously		
limited minimization	***pinon***	
f(x,y,z) with smallest i between 0 and x where ternary function h(i,y,z)=0 (value x+1 if there is no value 0) with ***Λ1*** for function h.	***Λzllimi Λ1Λzrtlimi***	3
f(x,y,z) with smallest i between 0 and function-limit g(x,y,z) given by ***Λ2*** where ternary function h(i,y,z)=0 (value g(x,y,z)+1 if there is no value 0) with ***Λ1*** for function h.	***Λzllimi Λ1Λzctflimi Λ2Λzrtflimi***	3
f(x,y) with smallest i between 0 and x where binary function h(i,y)=0 (value x+1 if there is no value 0) with ***Λ1*** for function h.	***Λzllimi Λ1Λzrblimi***	2
f(x) with smallest i between 0 and x where h(i,x)=0 with variable x	***Λzllimi Λ1Λzruvlimi***	1
f(x) gives the smallest value of i given by between 0 and function-limit g(x) given by ***Λ2*** where h(i,x)=0 if there is no zero the value is put to x+1 ; h(i,x) is given by ***Λ1***	***Λzllimi Λ1 Λzcuflimi Λ2 Λzruflisp***	1
f(x,y) with smallest i between 0 and x where ternary function h(i,x,y)=0 with variable x (value x+1 if there is no value 0) with ***Λ1*** for function h.	***Λzllimi Λ1Λzrbvlimi***	2
limited maximization		
f(x,y) with highest i between 0 and x where binary function h(i,y)=0 with variable x, (value x+1 if there is no value 0) with ***Λ1*** for function h.	***ΛzllimaΛ1Λzrbvlima***	2
other limited minimizations of higher arity analogously		
denumeration for characteristic	***pinon***	
auxilary recursion function h(x) that is the limited minimization of c(i)=0 and x<i, where c(i) is a characteristic function given by ***Λ1*** and the appearance of a zero is guaranteed by a majorant m(x) given by ***Λ2***	***Λxrecdenu*** = ***Λzllimi***{***Λcjy Λ1 Λmay***} ***ΛzcuflimiΛ2 Λzruflisp***	1
denumeration function for the zeros of characteristic function c(i) with the majorant m(x). The first zero is given by argument value 1 . It is obtained with the above unary recursion function h(x) with recursion start 0 .	20***Λxrecdenu*** = ***Λzldenu Λ1 Λzcdenu Λ2 Λzruflisp***	1

Table C6.2.4 Programming with limits(continuation)

The following table makes use of programming with limits:

		<u>primality, usual method</u>		
Λxprim	Λxprim	auxiliary for primality: count of divisors of x up to y-1	20{***ΛaddΛupr***{***ΛidivyΛtprΛbpr***}}	2
Λprimy	Λprimy	primality	{***Λsgy***{***Λudc***{***ΛxprimΛuprΛupr***}}}	1
Λcompy	Λcompy	composity(nonprime, not 0 1)	{***Λngy***{***ΛdjyΛprimyΛudc***}}	1
		<u>use of twofold limited quantive</u>		
Λxprima	Λxprima	auxiliary for alternative primality	{***Λdjy***{***ΛieqyΛmulΛtpr***}{***ΛdjyΛuqy***{***ΛuqyΛbpr***}}}	3
Λprimay	Λprimay	alternative for primality	{***ΛcjyΛugyΛzliomΛzliom ΛxprimaΛzrualispΛzrualisp***}	1
		<u>use of pairs</u>		
Λxprimaa	Λxprimaa	auxiliary for alternative primality	2{10}{***ΛmulΛupr***{***ΛadiΛtpr***{***Λmul***{***Λmul***{***ΛadcolΛbpr***}{***Λsgy***{***Λudc***{***ΛadcolΛbpr***}}}}{***Λmul***{***ΛadrowΛbpr***}{***Λsgy***{***Λudc***{***ΛadrowΛbpr***}}}}}}}	2
Λprimaay	Λprimaay	alternative for primality	{***ΛcjyΛugy***{***Λngy***{***ΛxprimaaΛuprΛbpo***}}}	1
		<u>application of denumeration</u>		
Λprime	Λprime	$f_{prime}(x)$ 0,2,3,5,7,11,… majorant x!+1	***ΛzIdenu Λprimy Λcdenu*** {1***Λfact***}***Λzruflisp***	1
Λcmjdivy	Λcmjdivy	auxiliary common divisibility y and z divisible by x	{***Λcjy***{***ΛdivyΛbprΛupr***}{***ΛdivyΛtprΛupr***}}	3
Λgrcmdi	Λgrcmdi	greatest common divisor	***ΛzllimaΛcmjdivy Λzrbvlima***	2
Λcoprimy	Λcoprimy	coprimality	{***ΛuqyΛgrcodi***}	2
Λcmdivy	Λcmdivy	auxiliary common divisibility x divisible by y and z	{***ΛcjyΛdivy***{***ΛdivyΛuprΛtpr***}}	3
Λlecmmu	Λlecmmu	least common multiple	***Λzllimi Λcmdivy Λzrbvlimi***	2
Λppsdec	Λppsdec	prime-power suite decode ppsdec(x,y) code x, position y, arity z $x=2^{f(0)}\ 3^{f(1)}\ 5^{f(2)} \ldots$ $f_{prime}(y+1)^{f(y)}\ y<z$	***Λzllima***{***ΛdivyΛbpr***{***Λexp***{***ΛprimeΛtpr***}***Λupr***}}} ***Λzrbvlima***	2
Λadsdec	Λadsdec	antidiagonal suite decode adsdec(x,y,z) code x, position y, arity z x = pair(…pair(pair(f(0),f(1)),f(2)),…,f(z))	*do not want to be buggered*	3
Λppssdec	Λppssdec	prime-power succession suite decode ppssdec(x,y) code x, position y, arity z $x=2^{f(0)+1}\ 3^{f(1)+1}\ 5^{f(2)+1} \ldots$ $f_{prime}(y+1)^{f(y)+1}\ y<z$		
Λppssari	Λppssari	prime-power succession suite arity		

Table C6.2.5 Prime numbers and suite coding (to be continued)

Λadpairs	Λadpairs	antidiagonal pair succession code pairs(x,y)	{1***Λadpair***}	3
Λadssdec	Λadssdec	antidiagonal succession suite decode adssdec(x,y) arity is included code x, position y y = pairs(…pairs(pairs(f(0),f(1)),f(2)),…,f(z))		3
Λadssari	Λadssari	antidiagonal pair succession suite arity		3
Λgbeta	Λgbeta	Gödel beta-function suite decoding gbeta(x,y,z)=dir (*x*,y(z+1)+1) position x, dividend code y, divisor code z, arity u suite f(0),f(1),f(2),..,f(u) f(z)= gbeta(x,y,z) [1)] z<u+1	{***ΛdirΛupr***{1{***ΛmulΛbpr***{1***Λtpr***}}}}	3
Λbgbeta	Λbgbeta	binary Gödel beta-function suite decoding position x, code y, arity u bgbeta(x,y)= gbeta(x, adrow(y),adcol(y)) no including of arity coding as in prime power and antidiagonal suite coding	{***ΛdirΛupr***{1{***Λmul***{***ΛadrowΛbpr***}{1{***ΛadcolΛbpr***} }}}}	3
Λobezy	Λobezy	ordered Bézout quadruple		4

[1)] *as opposed to prime-power suite and antidiagonal coding there is no straightforward method to find the Gödel beta-code for a suite*

Table C6.2.5 Prime numbers and suite coding (continuation)

Λfcg	Λfcg	fication generator	20{***Λdsa*** {***Λdsa Λoufc Λupr*** }***Λvfc*** }	1
		Λprg(Λ_1)()=Λ_1		
		Λprg(Λ_1)(Λ_2)=Λ_1		
		Λfcg(0)=0	0()=0	
		Λfcg(1)={10}	{10}()=1	
		Λfcg(2)={1{10}}	{1{10}}()=2	
		Λfcg(3)={1{1{10}}}	{1{1{10}}}()=3	
		…		
Λprg	Λprg	projection generator	{***Λmul Λsgy*** {2***Λbnufc***{***Λdsa*** {***Λdsa Λbfc Λupr*** }***Λbnufc*** }***Λudc*** }}	1
		Λprg(0)()=0	0	
		Λprg(1)(Λ_1)=Λ_1	201	
		Λprg(1)(Λ_1;Λ_2)=Λ_2	2201201	
		Λprg(3)(Λ_1;Λ_2;Λ_3)=Λ_3	22201201201	
		…		
Λcsg	Λcsg	cession generator (with a little "by cases")	{***Λadd*** {***Λmul Λngy Λbnufc*** }{***Λmul Λsgy*** {2{10}{***Λdsa*** {***Λdsa Λoufc Λupr*** }***Λvfc*** }}***Λudc*** }}}	1
		Λcsg(0)(Λ_1)=(0+Λ_1)	201	
		Λcsg(1)(Λ_1)=(1+Λ_1)	1	
		Λcsg(2)(Λ_1)=(2+Λ_1)	{11}	
		Λcsg(3)(Λ_1)=(3+Λ_1)	{1{11}}	
		…		
Λpcg	Λpcg	plication generator	{***ΛdsaΛbnfcΛcsg***}	1
		Λpcg(0)(Λ_1)=0	0	
		Λprg(1)(Λ_1)=Λ_1	201	
		Λprg(2)(Λ_1)=(2×Λ_1)	20{11}	
		Λprg(3)(Λ_1)=(3×Λ_1)	20{1{1{11}}}	
		…		
		<u>modified-Ackermann-function</u>		
		ΛMACK(Λ;Λ)		
Λbpc	Λbpc	duplication	20{11}	
Λbpt	Λbpt	bi-ponentiation	2{10}20{11}	
Λbspt	Λbspt	bi-superponentiation	2{10}2{10}20{11}	
Λbsspt	Λbsspt	bi-supersuperponentiation	2{10}2{10}2{10}20{11}	
		…		
Λmackg	Λmackg	function generator	2***Λbnouuvfc*** {***ΛdsaΛbounvfc Λupr*** }	1
		Λmackg(0) =Λbpc	20***Λbcs***	
		Λmackg(1)=Λbpt	2{10}***Λbpc***	
		Λmackg(2)=Λbspt	2{10}***Λbpt***	
		Λmackg(3)=Λbsspt	2{10}***Λbspt***	
		…		
		<u>non-primcursive function</u>		
ΛMACK(Λ_1;Λ_2)		Λmackg(Λ_1)(Λ_2)		

Table C6.2.6 Generator technique and non-primcursive functions

metafunctum	**pinon**	*functum (function or relation)*	*arity*
		functions	
(Λ*Λ)	Λtv-sa	*novitrigintal synaption*	2
(Λ∂)	Λtv-ssdel	*novitrigintal subscript deletion*	1
(Λ∂Λ)	Λtv-chdel	*novitrigintal character deletion*	1
(Λ;Λ∫Λ)	Λtv-repl	*novitrigintal replacement*	3
Λ ◊(Λ;Λ)	Λtv-rese	*novitrigintal free arity*	1
Λ ♦(Λ;Λ)	Λtv-book	*novitrigintal bound arity*	1
Λ′(Λ)	Λtv-succ	*novitrigintal succession* [1)]	1
Λ+(Λ)	Λtv-pnsucc	*novitrigintal petit-number succession* [2)]	1
Λcharl(Λ)	Λtv-length	*novitrigintal character-length* [2)]	1
Λcharc(Λ;Λ)	Λtv-charcount	*novitrigintal character-count* [2))]	2
Λcharp(Λ;Λ;Λ)	Λtv-charprof	*novitrigintal character-projection* [2)]	3
ϕ−(ϕ)	***ϕ⊕(ϕ)***	***ϕ©(ϕ)***	
ϕ+(ϕ;ϕ)	***ϕ⊕(ϕ;ϕ)***		
ϕ−(ϕ;ϕ)	***ϕ⊕(ϕ;ϕ;ϕ)***	*multary metarelations*	
Λ≈Λ	Λtv-apty	*novitrigintal aptity*	2
Λ≺Λ	Λtv-breviory	*novitrigintal breviority*	2
Λ⊃Λ	Λtv-suitconty	*novitrigintal suitable containment*	2
Λ/Λ	Λtv-boundconty	*novitrigintal bound containment*	2
Λ⅃Λ	Λtv-freeconty	*novitrigintal free containment*	2
Λ~Λ	Λtv-comply	*novitrigintal compatability*	2
Λ<Λ	Λtv-miy	*novitrigintal minority* [1)]	2
Λ⇒Λ	Λtv-uinfy	*novitrigintal unary inference*	2
Λ;Λ⇒Λ	Λtv-binfy	*novitrigintal binary inference*	3
		metaproperty examples	1
zero(Λ)	Λtv-zeroy	*novitrigintal zero characteristic*	
capital-greek-letter(Λ)	Λtv-capital-greek-lettery	*novitrigintal capital-Greek-letter characteristic*	
capital- latin-word(Λ)	Λtv-capital-latin-wordy	*novitrigintal capital-Latin-word characteristic*	
sentence(Λ)	Λtv-sentency	*novitrigintal sentence characteristic*	
truth(Λ)	Λtv-truthy	*small **truth** (as opposed to capital **TRUTH**)*	
⇒Λ** or **tautiom(Λ)	Λtv-ninfy *or*Λtv-tautiomy	*novitrigintal nullary inference or novitrigintal tautiom characteristic*	
		metarelation	
derivation(Λ;Λ)	Λtv-derivationy	*novitrigintal segment-sentence derivation characteristic*	2

[1)] in addition to synaption full succession is needed [2)] in addition to synaption count-succession is needed

*Table C6.2.7 Representing metafuncta as primcursive functa via **pinon** strings*

Gödel translation $Λ⇑(Λ)$ maps metaindivual novitrigintals to individual decimals, backward Gödel cislation $Λ⇓(Λ)$. This induces functa from metafuncta, e.g.

$Λ⇑((Λ_1 * Λ_2)) = \text{Λtv-sa}(Λ⇑(Λ_1);Λ⇑(Λ_2))$ $Λ⇓(\text{Λtv-sa}(Λ⇑(Λ_1);Λ⇑(Λ_2))) = (Λ_1 * Λ_2)$

$Λ⇑((∂Λ_1)) = \text{Λtv-del}(Λ⇑(Λ_1))$ $Λ⇓(\text{Λtv-del}(Λ⇑(Λ_1))) = (∂Λ_1)$

$$∀Λ_1[[\textbf{sentence}(Λ_1)] ↔ [\textbf{Truth}(\text{Λtv-sentency}(Λ⇑(Λ_1))=0)]]$$

truth:: syniom ¦ aponom ¦ basiom ¦ tautiom ¦ haplonom ¦ zygonom

<u>initial:</u>		
auxiliary	x -	for auxiliary ***pinon-constant*** and ***spinon-constant*** strings resp.
auxiliary	z	for auxiliary ***number-constant*** that are not ***pinon-*** or ***spinon-constant*** strings
inverse	w	
non-, un-, in-	i	
transposed	j	transposition of binary argument
remainder	r	in connection with inverses w and others
goedelisation	g	
number mnemos	n u b t q p s h o v d (not un) du ... dv bn tn qn pn sn hn on vn unn unnn ...	
<u>exitial:</u>		
alternative	a ay	
generator	g	generates a ***pinon*** string
metarepresentation	m, me	of metafuncta
characteristic	y	only values 0 and 1 for 'true' and 'false', i.e. characteristic function
<u>for auxiliary</u> after z	l c r	left center right part i k e f other identifications

<u>*with number mnemo unary and one nullary*</u>		
cession	cs	addition, with fixed summand
decession	dc	truncated subtraction, with fixed subtrahend
fication	fc	constantion (only nullary)
entire lorithmation	lr	entire logarithmation, with fixed base
plication	pc	multiplication, with fixed factor
power, potention	po	exponentiation, with fixed exponent
entire rooting	rt	entire radication, with fixed root exponent
entire section	sc	entire division, with fixed divisor
ponentiation	pt	exponentiation, with fixed base
cession	cs	addition, with fixed summand
equality inequality	qy gy	with fixed value unary
entire lorithmability	lry	
entire rootability	rty	
entire sectivity	scy	
fibonacci	fibo	
<u>*binary*</u>		
ladder function	la	
synaption	sa	
deletion	del	
<u>*ternary*</u>		
replacement	rep	
<u>*multary*</u>		
multary addition	mad	
antidiagonal tuple	adsec	suite coding
<u>*all arities*</u>		
pair-addition	pad	
projection	pr	
maximum	max	
minimum	min	

<u>*no number mnemo ternary quaternary*</u>	
pair suite dec.	adsdec
Bézout quadruple	bezouty
Gödel's betafunction	gbeta
modulo-congruity	modcgy
prime-power suite dec	ppsdec

<u>*no number mnemo unary*</u>		
carlation	carl	
factorial	fact	
composity	compy	
evenness	evy	
negation	ngy	nullum-inequality
oddity	ody	
pinity	piny	
primality	primy	
signation	sgy	nullum-equality nqy
<u>*no number mnemo binary*</u>		
addition	add	
absolute difference	adi	
antidiagonal row	adrow	decoding
antidiagonal column	adcol	decoding
antidiagonal pair	adpair	coding
addition succession	ads	
bicondition characteristic	bcy	
congruity	cgy	
conjunction characteristic	cjy	
coprimality	coprimy	
entire division, remainder	div dir	
entire divisibility	divy	
disjunction characteristic	djy	
equal-majority	emay	
equal-minority	emiy	
equality	eqy	
exponentiation	exp	
greatest common divisor	grcmdi	
implication characteristic	imy	
least common multiple	lecmmu	
entire logarithmation	log	
majority	may	
minority	miy	
multiplication	mul	
entire radication	rad	
truncated subtraction	sub	
supplication	sup	(x+1)y
super-exponentiation	sexp	
supersuper-exponentiation	ssexp	

Table C6.3 Mnemonic rules for **descriptor** *of* **pinon** *and* **number** *strings (to be continued)*

		***number** part for*
inv	inv	entire inverse
lisu	lisu	limited sum
lipr	lipr	limited product
lisp	lisp	limited sum and product
flisp	flisp	function-limited sum and product
liom	lom	limited omnitive
lien	len	limited entitive
liqu	lqu	limited quantive
fliom	flom	function-limited omnitive
flien	flen	function-limited entitive
fliqu	flqu	function-limited quantive
limi	limi	limited minimization
flimi	flimi	function-limited minimization
lima	lima	limited maximization
denu	denu	denumeration
		in combinations
a	a	absolute
ad	ad	addition, antidiagoanl pair
aut	aut	auto
cm	cm	common
di	di	divisor, difference
e	e	equal
en	en	entitive
f	f	function
gr	gr	greatest
le	le	least
li	li	linited
ma	ma	major
mi	mi	minor
mod	mod	modulo
mu	mu	multiple
om	om	omnitive
pr	pr	product
qu	qu	quantive
sp	sp	sum and product
su	su	super, sum

*Table C6.3 Mnemonic rules for **descriptor** of **pinon** and **number** strings (continuation)*

C7 Boojum-function and Snark-function

The **Boojum-function** (Λ□Λ) and the **Snark-function** (□Λ) were put forward in the publication ***The Snark, a counterexample to Church's thesis?*** as pretty weird calculative functions. They can be defined for concrete calcule LAMBDA of decimal pinitive arithmetic using the FUME-method that contains two tiers of precise languages: object-language Funcish and metalanguage Mencish. Now the lowest values and some more insight into these functions are given.

One has to do it in a way that looks a little queer. But one has to avoid problems with leading 0 that arise with concatenation. The following table gives the Boojum-function (Λ□Λ) where the rows refer to the first position and Λ_1 to the second position. Column 6 gives the values of the Snark-function (□Λ) with respect to input of the Gödel-number in column 4. The ***norm-unary-formula*** strings are

formed from 13 digits	; ( ) 0 1 2 3 4 5 6 7 8 9 *and* Λ_1 *(at least one appearance of* Λ_1*)*
with 14 numbers (col.5)	0 1 2 3 4 5 6 7 8 9 10 11 12 13 , *map to* ***quadro-decimal*** *strings*
use ***quadro-decimal*** *digits*	0 1 2 3 4 5 6 7 8 9 A B C D , *map to* ***decimal*** *strings (column 4)*
with lexicographic order …	1,14,196,2744,38416,537824,7529536,105413504,1475789056
conventionally written	1_{14}, 10_{14}, 100_{14}, 1000_{14}, 10000_{14}, 100000_{14}, 1000000_{14}, 10000000_{14} …

not ***norm-unary-formula*** *formed with the 14 digits* [1)]		*lengtth*	**decimal** *Gödel-number*	**quadro-decimal**	*Snark-function* (□Λ) *of Gödel-numbe*
examples					*all* =1
;		1	0	0	(□0)=1
(		1	1	1	(□1)=1
)		1	2	2	(□2)=1
(;;;		4	2744	1000	(□2744)=1
6789		4	26822	9ABC	(□26822)=1
202(Λ_1)		6	2818468	5351D2	(□2818468)=1
norm-unary-formula	*processive*				
Λ_1		1	13	D	(□13)=14
0(Λ_1)		4	8612	31D2	(□8612)=1
1(Λ_1) *succession*		4	11356	41D2	011358
Λ_1(0)	yes	4		D132	1
Λ_1(1)	yes	4		D142	1
Λ_1(2)	yes	4		D152	1
...	yes				1
Λ_1(9)	yes	4		D1C2	1
Λ_1(Λ_1)	yes	4		D1D2	1
Λ_1(10)	yes	5		D1432	1
...	yes				1
Λ_1(99)	yes	5		D1CC2	1
0(0;Λ_1)		6		3130D2	1
0(1;Λ_1)		6		3140D2	1
...					1
0(9;Λ_1)		6		31C0D2	1
0(Λ_1;0)		6		31D032	1
0(Λ_1;1)		6		31D042	1
...					1
0(Λ_1;9)		6		31D0C2	1
0(Λ_1;Λ_1)		6		31D0D2	1

1(0;Λ1)		6		4130D2	2
1(1;Λ1)		6		4140D2	3
...					
1(9;Λ1)		6		41C0D2	11
1(Λ1;0)		6		41D032	2225428
1(Λ1;1)		6		41D042	2225442
...					
1(Λ1;9)		6		41D0C2	2225582
1(Λ1;Λ1)		6		41D0D2	2225442
201(Λ1) *unary projection*[2)]		6	2815724	5341D2	2815725
Λ1(0;Λ1)	yes	6		D130D2	1
Λ1(1;Λ1)	yes	6		D140D2	1
...	yes				
Λ1(9;Λ1)	yes	6		D1C0D2	1
Λ1(Λ1;0)	yes	6		D1D032	1
Λ1(Λ1;1)	yes	6		D1D042	1
...	yes				1
Λ1(Λ1;9)	yes	6		D1D0C2	1
Λ1(Λ1;Λ1)	yes	6		D1D0D2	1
no more consecutive					
0(Λ1)(Λ1)	yes	7		31D21D2	1
1(Λ1)(Λ1)	yes	7	31161244	41D21D2	1
Λ1(Λ1)(Λ1)	yes	7		D1D21D2	1
201(Λ1)(Λ1)	yes	9		5341D21D2	1
all primiive recursive functions are trivial e.g.					
22011(Λ1;Λ1) *x+x* [2)]		10	553441D0D2 *translate*	553441D0D2	AA6883C1C4 *translate to decimal*
22011(13;Λ1) *13+x* [2)]		11	553441460D2 *translate*	553441460D2	55344146104 *translate to decimal*
and now its getting really wild					
diagonal modified Ackermann [2)]					*applied to its Gödel numberr*
Λmackg(Λ1)(Λ1)	yes		***Λmackg***1D21D2 *translate, very long*	***Λmackg***1D21D2 *very long*	*incredibly incredible*
but that's not the end of incredibilities					

[1)] for strings that are not formed for the 14 digits the corresppnding Gödel-dedq-translation ***Λ⇑⇑(Λ)*** gives result 0
for strings that are not decimal number the corresppnding Gödel-dedeq-cinslation ***Λ⇓⇓(Λ)*** gives result 0

[2)] see publication ***Programming pinon strings in concrete calcule*** LAMBDA