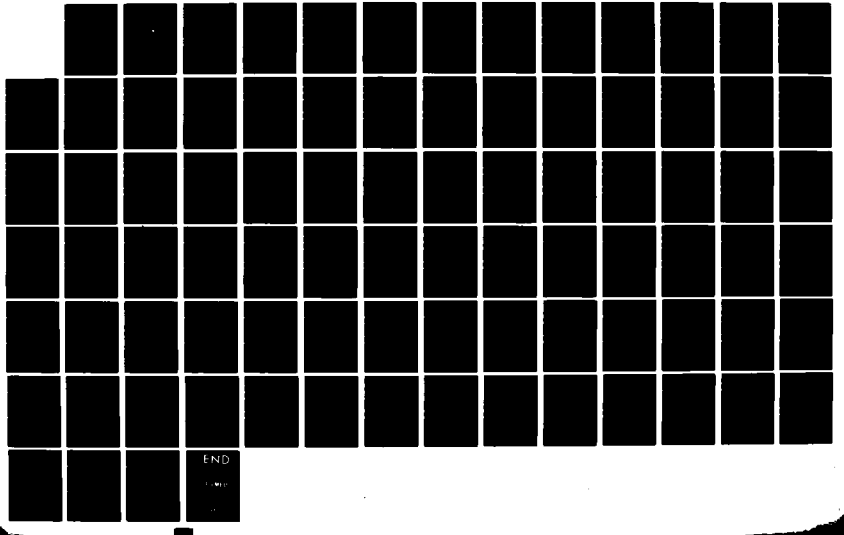
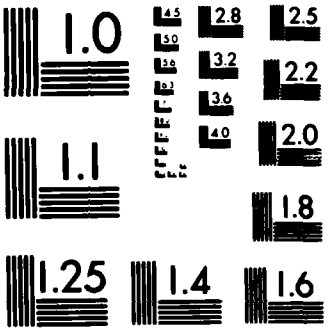
AD-A149 079 COMPUTABLE REAL ANALYSIS(U) NAVAL POSTGRADUATE SCHOOL 1/1
MONTEREY CA B J MACLENNAN DEC 84 NPS52-84-024

UNCLASSIFIED F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NPS52-84-024

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A149 079

DTIC
ELECTE
JAN 9 1985
S
A

COMPUTABLE REAL ANALYSIS

BRUCE J. MacLENNAN

December 1984

DTIC FILE COPY

Approved for public release; distribution unlimited.

Prepared for: Chief of Naval Research
Arlington, VA 22217

84 12 31 071

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Commodore R. H. Shumaker
Superintendent

D. A. Schrady
Provost

The work reported herein was supported by Contract N00014-84-WR-24087 from the Office of Naval Research.

Reproduction of all or part of this report is authorized.

This report was prepared by:

BRUCE J. MacLENNAN
Associate Professor and Acting
Chairman of Computer Science

Reviewed by:

BRUCE J. MACLENNAN
Acting Chairman
Department of Computer Science

Released by:

KNEALE T. MARSHALL
Dean of Information and
Policy Science

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS52-84-024	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Computable Real Analysis		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Bruce J. MacLennan		8. CONTRACT OR GRANT NUMBER(s) N00014-84-WR-24087
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Chief of Naval Research Arlington, VA 22217		12. REPORT DATE December 1984
		13. NUMBER OF PAGES 75
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Real analysis, real number system, constructive mathematics, recursive function theory, lambda calculus, computable numbers, computability theory, definition of real number, foundations of mathematics, functional programming, finitistic mathematics, computable real analysis.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present a model of the real numbers that is completely finitistic. Every real number is represented by a finite structure - specifically, a finite sequence of symbols from a finite alphabet. All of the arithmetic operations on reals are also finite and can be evaluated on a computer. We allow nothing that cannot be described by a finite algorithm - whether numbers or sequences or functions. This development is carried through the fundamental theorem of of calculus.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

COMPUTABLE REAL ANALYSIS

B. J. MacLennan
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

CONTENTS

Abstract: 1

1. Introduction 1

2. The Lambda Calculus 3

2.1 Syntax 3
- 2.1.1 Tokens: 3
- 2.1.2 *Abstractions*: 3
- 2.1.3 Applications: 3
- 2.1.4 Exclusion: 4
- 2.1.5 Definition of Bound Identifier: 4
- 2.1.6 Definition of Free Identifier: 4

2.2 Semantics 4
- 2.2.1 Substitution Notation: 4
- 2.2.2 Renaming Rule: 4
- 2.2.3 Substitution Rule: 4
- 2.2.4 Definition of Reduction: 4
- 2.2.5 Definition of Normal Reduction Order: 5
- 2.2.6 Definition of Normal Form: 5
- 2.2.7 Church-Rosser Theorem: 5
- 2.2.8 Definition of a Formula Being Defined: 5
- 2.2.9 Universality of Normal Reduction Order: 5
- 2.2.10 Definition of Convertibility: 5

3. Basic Definitions 6

3.1 Parameters 6
- 3.1.1 Dyadic Functions: 6
- 3.1.2 N-adic Functions: 6
- 3.1.3 Omission of Parentheses (Applications): 6
- 3.1.4 Omission of Parentheses (Abstractions): 6
- 3.1.5 *Alternative Brackets*: 6

3.2 Combinators 6
- 3.2.1 Identity: 6
- 3.2.2 Constant: 6
- 3.2.3 Projection Functions: 6
- 3.2.4 Monadic Compositor: 7
- 3.2.5 Monadic Formalizer: 7
- 3.2.6 Dyadic Compositor: 7
- 3.2.7 Dyadic Formalizer: 8
- 3.2.8 Dual Compositor: 8
- 3.2.9 Elementary Permutator (Conversion): 8
- 3.2.10 S Combinator: 9
- 3.2.11 Psi Combinator: 9
- 3.2.12 Paradoxical Combinator: 9
- 3.2.13 Recursive Definition: 9

A-1

DTIC COPY INSPECTED

3.2.14 Seminormal Form: 10

3.3 Conditionals 10
3.3.1 True Value: 10
3.3.2 False Value: 11
3.3.3 Definition of Boolean Value: 11
3.3.4 Negation: 11
3.3.5 Conjunction: 11
3.3.6 Disjunction: 11
3.3.7 Equivalence: 11
3.3.8 Dyadic Conditional: 12
3.3.9 N-adic Conditional: 12

3.4 Lists 12
3.4.1 First of Three: 12
3.4.2 Second of Three: 12
3.4.3 Third of Three: 12
3.4.4 List Value: 12
3.4.5 Null List: 13
3.4.6 List Construction: 13
3.4.7 Null Test: 13
3.4.8 Head of List: 13
3.4.9 Tail of List: 13
3.4.10 List Notation: 14
3.4.11 Pair List: 14
3.4.12 Definition of List: 14
3.4.13 Example Normal Form: 14
3.4.14 Nonstandard Objects: 15

4. Natural Numbers 16

4.1 Primitive Ideas 16
4.1.1 Definition of Natural Number: 16
4.1.2 Numeric Denotations: 16
4.1.3 Example Normal Form: 16
4.1.4 Nonstandard Objects: 16
4.1.5 Zero Predicate: 17
4.1.6 Successor: 17
4.1.7 Predecessor: 17

4.2 Arithmetic 17
4.2.1 Repeated Composition: 17
4.2.2 Sum: 18
4.2.3 Difference: 18
4.2.4 Product: 18

4.3 Predicates and Relations 18
4.3.1 Less-Than Relation: 18
4.3.2 Greater-Than Relation: 18
4.3.3 Inequality Relation: 19
4.3.4 Equality Relation: 19
4.3.5 Less-or-Equal Relation: 19
4.3.6 Greater-or-Equal Relation: 19

4.4 Miscellaneous Operations 19
4.4.1 *Quotient:* 19
4.4.2 Factorial: 19
4.4.3 Definition of Sequence: 19
4.4.4 Subscript Notation: 20

5. Integers 21

5.1 Primitive Ideas 21
5.1.1 Definition of Integer: 21
5.1.2 Attach Plus-sign: 21
5.1.3 Attach Minus-sign: 21
5.1.4 Denotations: 21
5.1.5 Example Normal Form: 21
5.1.6 Nonstandard Objects: 22
5.1.7 Extract Magnitude: 22

5.2 Primitive Predicates 22
5.2.1 Minus-sign Test: 22
5.2.2 Plus-sign Test: 22
5.2.3 Zero Test: 22

5.3 Primitive Operations 23
5.3.1 Absolute Value: 23
5.3.2 Negate: 23
5.3.3 Successor: 23
5.3.4 Predecessor: 23
5.3.5 Sum: 23
5.3.6 Difference: 24

5.4 Predicates and Relations 24
5.4.1 Equality: 24
5.4.2 Negative Predicate: 24
5.4.3 Positive Predicate: 24
5.4.4 Less-Than Relation: 24
5.4.5 Greater-Than Relation: 24
5.4.6 Greater-or-Equal Relation: 25
5.4.7 Less-or-Equal Relation: 25
5.4.8 Maximum: 25

5.5 Multiplicative Operations 25
5.5.1 Product: 25
5.5.2 Quotient: 25

6. Rational Numbers 26

6.1 Primitive Ideas 26
6.1.1 Definition of Rational Number: 26
6.1.2 Construction: 26
6.1.3 Notation: 26
6.1.4 Denotations: 26
6.1.5 Example Normal Form: 26
6.1.6 Nonstandard Objects: 27
6.1.7 Extract Numerator: 27
6.1.8 Extract Denominator: 28

6.2 Arithmetic Operations 28
6.2.1 Sum: 28
6.2.2 Absolute Value: 29
6.2.3 Negation: 29
6.2.4 Difference: 29
6.2.5 Product: 29
6.2.6 Reciprocal: 30
6.2.7 Quotient: 30
6.2.8 Floor: 30
6.2.9 Ceiling: 31

6.3 Predicates and Relations 31
6.3.1 Nonnegative: 31
6.3.2 Negative: 31
6.3.3 Less-Than: 31
6.3.4 Greater-Than: 31
6.3.5 Less or Equal: 32
6.3.6 Greater or Equal: 32
6.3.7 Equality: 32
6.3.8 Maximum: 32

6.4 Operations on Nonstandard Rational Numbers 32
6.4.1 Infinite and Indefinite Rational Numbers: 32
6.4.2 Predicates: 33
6.4.3 Standard Equality: 33
6.4.4 Nonstandard Equality: 33
6.4.5 Nonnegative: 34
6.4.6 Negative: 34
6.4.7 Nonstandard Addition: 34
6.4.8 Negation: 35
6.4.9 Difference: 35
6.4.10 Ordering: 35
6.4.11 Products, Reciprocals and Quotients: 36

7. Real Numbers 37

7.1 Primitive Ideas 37
7.1.1 Definition of a Regular Sequence: 37
7.1.2 Definition of Real Number: 37
7.1.3 Equality: 37
7.1.4 Nonstandard Objects: 38
7.1.5 Creation of Reals from Rationals: 38
7.1.6 Real Denotations: 38
7.1.7 Example Normal Form: 39
7.1.8 Example Normal Form of Irrational Number: 39
7.1.9 Rational Approximation: 40

7.2 Arithmetic 40
7.2.1 Sum: 40
7.2.2 Maximum: 41
7.2.3 Negation: 41
7.2.4 Minimum: 41
7.2.5 Absolute Value: 41
7.2.6 Canonical Bound: 41
7.2.7 Product: 42
7.2.8 Exponentiation to Natural Power: 42
7.2.9 Notation: 42

7.3 Predicates and Relations 43
7.3.1 Positive with Modulus: 43
7.3.2 Positive: 43
7.3.3 Negative with Modulus: 43
7.3.4 Zero with Modulus Predicate: 43
7.3.5 Equality with Modulus: 44
7.3.6 Nonnegative: 44
7.3.7 Greater-Than with Modulus: 44
7.3.8 Greater-Than Relation: 44
7.3.9 Less-Than with Modulus: 45
7.3.10 Less-Than Relation: 45

7.3.11 Greater-or-Equal with Modulus: 45
7.3.12 Greater-or-Equal Relation: 45
7.3.13 Less-or-Equal with Modulus: 45
7.3.14 Less-or-Equal Relation: 45
7.3.15 Inequality with Modulus: 45
7.3.16 Inequality Relation: 46

7.4 Reciprocals and Quotients 46
7.4.1 Reciprocal: 46
7.4.2 Zero Bound: 46
7.4.3 Reciprocal with Modulus: 46
7.4.4 Quotient with Modulus: 46

7.5 Important Properties of the Real Numbers 47
7.5.1 Approximation Theorem: 47
7.5.2 Rational Between Two Reals: 47
7.5.3 Denseness of Rational Numbers: 47
7.5.4 Cantor's Theorem: 48

8. Calculus 52

8.1 Convergence, Limits and Series 52
8.1.1 Convergence: 52
8.1.2 Limit: 52
8.1.3 Convergence to Limits: 53
8.1.4 Properties of Limits: 54
8.1.5 Compact Interval: 55
8.1.6 Compact Interval with Modulus: 55
8.1.7 Notation for Limits: 55
8.1.8 Partial Sums of Series: 56
8.1.9 Infinite Sums of Series: 56
8.1.10 Exponential Function: 57

8.2 Continuous Functions 57
8.2.1 Continuity on a Compact Interval: 57
8.2.2 Continuity on an Arbitrary Interval: 57

8.3 Differentiation 57
8.3.1 Differentiability: 57
8.3.2 Derivative: 58
8.3.3 Differentiation Theorem: 58
8.3.4 Properties of the Derivative: 59
8.3.5 Repeated Differentiation: 59
8.3.6 Partial Differentiation: 60

8.4 Integration 60
8.4.1 Quadrature: 60
8.4.2 Integral: 60
8.4.3 Reversed Range of Integration: 60
8.4.4 Properties of Definite Integral: 60
8.4.5 Indefinite Integral: 61
8.4.6 Properties of Indefinite Integral: 61
8.4.7 Fundamental Theorem of Calculus: 61
8.4.8 Natural Logarithm: 61
8.4.9 Definite Integral of N-adic Function: 61
8.4.10 Indefinite Integral of N-adic Function: 62

Appendix 1: Seminormal Form of Square-Root of 2 63

Appendix 2: LISP Program to Compute Seminormal Form of Square-Root 2 70

Appendix 3: LISP Program to Compute Length of Seminormal Form of Square Root
2 73

COMPUTABLE REAL ANALYSIS

B. J. MacLennan

Abstract:

We present a model of the real numbers that is completely *finitistic*. Every real number is represented by a finite structure – specifically, a finite sequence of symbols from a finite alphabet. All of the arithmetic operations on reals are also finite and can be evaluated on a computer. We allow nothing that cannot be described by a finite algorithm – whether numbers or sequences or functions. This development is carried through the fundamental theorem of calculus.

1. Introduction

In this report we develop systematically the foundations of a theory of *computable real analysis*. This is based on the view that the only objects that it makes sense to talk about are *finite* objects, and that the only operations it makes sense to talk about are those that can be completed in a *finite* number of operations.

This approach does not exclude all use of notions of infinity. In particular, it permits *potential infinities*, provided the operation generating the infinity can be finitely described, or is itself the result of a process that can be finitely described, and so forth, so that it can be reduced to a finite process in a finite number of steps. For example, we will permit the use of infinite sequences, provided that their enumeration functions are finitely describable and finitely computable. We eschew all use of *actual infinities*, that is, all infinities that are irreducible to finite objects.

This approach is most easily accomplished by restricting our attention to finite objects and computable functions. In other words, we will restrict our attention to computer programs operating on computer data. Computer programs and data are both finite objects, but they provide a convenient means for dealing with potential infinities.

Notice that in this approach every real number is represented by a *finite structure* – that is, a finite string of symbols chosen from a finite alphabet. In particular, irrational numbers are represented finitely. This reduction to finite structures is accomplished by identifying real numbers with programs that com-

pute sequences of rational approximations to the intended real number.

Various operations and functions of real numbers (e.g. the extracting of limits of sequences of reals) are also defined as programs, which means that these operations and functions are computable in a finite number of operations. Thus we have the foundations of a completely *finitistic* theory of real analysis.

There are many models of computation that could be used in this development. There are several reasons why we have chosen to use *functional programming* as a basis for our development of the foundations of real analysis:

1. Functional programming is mathematical in its style. Thus it lends itself to the definition of mathematical ideas and is conducive to proofs of mathematical properties.

2. Functional programming provides high-level, powerful means for manipulating computable functions. This is especially helpful in defining operations on potentially infinite objects.

3. Functional programs can be straight-forwardly reduced to formulas in the lambda calculus, a well-known model of computation. This reduction founds real analysis on a small number of basic ideas.

A secondary goal of this report is to demonstrate the use of functional programming concepts in constructivist mathematics.

We have omitted many of the proofs that our definitions are correct, or that they satisfy expected properties (commutativity, associativity, etc.). In most cases these proofs are routine and will not be missed. However, we have frequently given *informal* demonstrations that our definitions are correct.

In this report we follow closely the development in Errett Bishop's *Foundations of Constructive Analysis* (McGraw-Hill, New York, 1967). Both the definitions and order of presentation have been based on his, although they have been adapted where necessary to the functional approach. Where we have omitted proofs, the reader should consult Bishop's book, since his proofs are usually easily adapted to our approach.

2. The Lambda Calculus

In this chapter we develop the syntax and semantics of the lambda calculus. The lambda calculus is a model of computation equivalent in power to the Turing Machine and other well-known models of computation. By the generally accepted definition of computability, the lambda calculus can compute any computable function.

2.1 Syntax

2.1.1 Tokens:

We assume a denumerable infinity ν of symbols:

$$\nu = \{a, b, c, \ldots, z, \text{aa}, \text{ab}, \ldots, A, B, \ldots, \text{succ}_N, \text{pred}_N, \ldots, \perp, \ldots\}$$

These tokens can be considered strings made of symbols chosen from an underlying finite alphabet of symbols.

A token from ν is a *formula* of the lambda calculus.

Comments: Tokens are used to denote constants and for the names of functions and operations and for the formal parameters of functions.

2.1.2 Abstractions:

An expression of the form $(\lambda \nu E)$, in which ν is a token and E is a formula of the lambda calculus, is a *formula* of the lambda calculus.

Comments: The intended meaning of $(\lambda \nu E)$ is the function that takes any value ν into E. For example, $(\lambda x\,(x + 1))$ is the successor function, since it takes any x into $x + 1$.

2.1.3 Applications:

An expression of the form $(E\ E')$, in which E and E' are formulas of the lambda calculus, is a *formula* of the lambda calculus.

Comments: The notation $(f\ x)$ denotes the application of the function f to the argument x. For example, $((\lambda x\,(x + 1))\ 2)$ denotes the application of the successor function $(\lambda x\,(x + 1))$ to the argument 2.

2.1.4 Exclusion:

The only formulas of the lambda calculus are those described in (1) - (3) above.

2.1.5 Definition of Bound Identifier:

A variable ν occurs *bound* in a formula E if and only if there is some formula F such that $(\lambda \nu F)$ is a subformula of E.

2.1.6 Definition of Free Identifier:

A variable occurs *free* in a formula if and only if it does not occur bound in that formula.

2.2 Semantics

2.2.1 Substitution Notation:

The notation '$E\{\nu \leftarrow A\}$' represents the result of substituting the formula A for all free occurrences of the identifier ν in formula E.

2.2.2 Renaming Rule:

If y does not occur free in E, then we can perform the following *reduction by renaming:*

$$(\lambda x E) \Rightarrow (\lambda y E\{x \leftarrow y\})$$

2.2.3 Substitution Rule:

We can perform the following *reduction by substitution:*

$$((\lambda x E)\ A) \Rightarrow E\{x \leftarrow A\}$$

provided that it does not result in any free variable of A becoming bound.

2.2.4 Definition of Reduction:

A *reduction* is a sequence of one or more applications of the renaming and substitution rules. We write $E \Rightarrow F$ to denote that there is a reduction from E to F. We also use this symbol to introduce abbreviations and explicit definitions. Note that this means that all abbreviations and definitions are in principle eliminatable.

2.2.5 Definition of Normal Reduction Order:

We say that a reduction is performed in *normal reduction order* if and only if the outermost reducible application is always reduced before any inner applications. That is, we never reduce a formula by substitution that is a subformula of an application that could be reduced by substitution.

2.2.6 Definition of Normal Form:

A formula is said to be in *normal form* if and only if the substitution rule cannot be applied to it, even after intervening applications of the renaming rule.

2.2.7 Church-Rosser Theorem:

If a formula has a normal form, then that normal form is unique (up to applications of the renaming rule). Thus, the normal form of a formula can be considered that formula's *value*.

2.2.8 Definition of a Formula Being Defined:

We say that a formula is *defined* if and only if it has a normal form. Note that by Turing's Halting Theorem it is not finitely decidable (i.e., algorithmically decidable) whether or not a formula is defined.

2.2.9 Universality of Normal Reduction Order:

If a formula is defined, then a normal order reduction will reach its normal form.

2.2.10 Definition of Convertibility:

We call two formulas E and F *convertible*, and write '$E \Leftrightarrow F$', if and only if there is a normal form N such that $E \Rightarrow N$ and $F \Rightarrow N$. That is, two formulas are convertible if and only if they have the same normal forms.

Note that if E and F are defined then $E \Leftrightarrow F$ is finitely decidable: simply reduce both E and F to their normal forms (which are finite formulas), and compare them. This comparison can be done finitely since the formulas are finite.

3. Basic Definitions

In this chapter we introduce a number of abbreviations and notations, which make the lambda calculus convenient to use as a functional programming language. These extensions include one atomic data type, Booleans, and one composite data type, LISP-style lists.

3.1 Parameters

3.1.1 Dyadic Functions: $(\lambda(xy)E) \Rightarrow (\lambda x(\lambda y E))$

3.1.2 N-adic Functions: $(\lambda(x_1 x_2 \cdots x_n)E) \Rightarrow (\lambda x_1(\lambda x_2(\cdots(\lambda x_n E)\cdots)))$

3.1.3 Omission of Parentheses (Applications):

Dyadic: $(f\ x\ y) \Rightarrow ((f\ x)\ y)$

N-adic: $(f\ x_1\ x_2\ \cdots\ x_n) \Rightarrow (\cdots((f\ x_1)\ x_2)\ \cdots\ x_n)$

3.1.4 Omission of Parentheses (Abstractions):

$\lambda x E \Rightarrow (\lambda x E)$

$\lambda(x_1, \ldots, x_n)E \Rightarrow (\lambda(x_1, \ldots, x_n)E)$

3.1.5 Alternative Brackets: Other forms of parentheses, such as $[-]$ and $\{-\}$, may be used in place of $(-)$ for readability.

3.2 Combinators

3.2.1 Identity: $\mathrm{Id} \Rightarrow \lambda xx$

Comments: Note that $\mathrm{Id}\ a \Rightarrow a$, for all a.

3.2.2 Constant: $\mathbf{K} \Rightarrow \lambda(kx)k$

Comments: The function '$\mathbf{K}\ c$' returns the value c for all arguments:

$$\mathbf{K}\ c\ a \Rightarrow \lambda(kx)k\ c\ a \Rightarrow c$$

3.2.3 Projection Functions:

$1\mathrm{st} \Rightarrow \lambda(xy)x$

2nd $\Rightarrow \lambda(xy)y$

Comments: Thus, 1st $a\ b \Rightarrow a$ and 2nd $a\ b \Rightarrow b$. These functions are usually used in conjunction with other combinators.

3.2.4 Monadic Compositor:

$\mathbf{B} \Rightarrow \lambda(fgx)|f\ (gx)|$

$f \cdot g \Rightarrow \mathbf{B}\ f\ g$

Comments: Both '**B**' (the prefix form) and '$\cdot$' (the infix form) compute the composition of two monadic functions. To see this note that

$$(\log \cdot \sin)a \Rightarrow \mathbf{B}\ \log\ \sin\ a \Rightarrow \log(\sin\ a)$$

3.2.5 Monadic Formalizer:

$\Phi \Rightarrow \lambda(fabx)|f\ (ax)(bx)|$

$f \cdot (a,b) \Rightarrow \lambda x|f\ (ax)(bx)|$

Comments: If 'sum' is the addition function, then 'sum $\cdot (f,g)$' represents the functional sum of f and g, since

$$\text{sum} \cdot (f,g)\ x \Rightarrow \text{sum}\ (f\ x)\ (g\ x)$$

For example, sum $\cdot$ (sin,cos) is a function to compute the sum of the sines and cosines of its argument:

$$\text{sum} \cdot (\sin,\cos)\ \theta \Rightarrow \text{sum}\ (\sin\ \theta)\ (\cos\ \theta)$$

Thus this operation feeds a single input through two monadic functions, and then feeds the outputs of these into a dyadic function.

3.2.6 Dyadic Compositor:

$\Gamma \Rightarrow \lambda(fg)\{\lambda(xy)|f\ (gxy)|\}$

$f:g \Rightarrow \Gamma\ f\ g$

Comments: For example,

$$(\log:\text{sum})\ a\ b \Rightarrow \lambda(xy)|\log(\text{sum}\ x\ y)|\ a\ b \Rightarrow \log(\text{sum}\ a\ b)$$

This composes a monadic function with the output of a dyadic function.

3.2.7 Dyadic Formalizer:

$\Upsilon \Rightarrow \lambda(fabzy)[f\ (azy)(bzy)]$

$f:(a,b) \Rightarrow \lambda(xy)[f\ (axy)(bxy)]$

Comments: This operation feeds two arguments to each of two dyadic functions, and then feeds the two results to a third dyadic function. Thus, if f and g are dyadic functions, then sum:(f,g) is the functional sum of f and g. For example, to compute $(x-y)(x+y)$ we can use the function prod:(dif,sum):

prod:(dif,sum) x y $\Rightarrow$ prod (dif x y) (sum x y)

3.2.8 Dual Compositor: $\Omega \Rightarrow \lambda(fghxy)[f\ (gx)(hy)]$

Comments: This preprocesses the two inputs to a dyadic function with two separate monadic functions. For example,

(Ω sum sin cos) a b $\Rightarrow$ $\lambda(fghxy)[f\ (gx)(hy)]$ sum sin cos a b $\Rightarrow$ sum (sin a) (cos b)

Thus, 'Ω sum sin cos' can be read "the sum of the sin of the first and the cosine of the second." This combination can also be expressed by the dyadic compositor and the projection functions:

sum:(sin • 1st, cos • 2nd)

To see this, observe

sum:(sin • 1st, cos • 2nd) a b $\Rightarrow$ sum (sin • 1st a) (cos • 2nd a b)

$\Rightarrow$ sum (sin[1st a b]) (cos[2nd a b])

$\Rightarrow$ sum (sin a) (sin b)

3.2.9 Elementary Permutator (Conversion): conv $\Rightarrow \lambda(fxy)(fyx)$

Comments: For example, if (dif x y) computes the difference of x and y, then (conv dif 1) is the predecessor function, since

(conv dif 1) a $\Rightarrow$ $\lambda(fxy)(fyx)$ dif 1 a $\Rightarrow$ dif a 1

Thus conv exchanges the arguments of a dyadic function.

3.2.10 S Combinator: $\mathbf{S} \Rightarrow \lambda(f\,g\,x)|f\,x\,(g\,x)|$

Comments: The S combinator is difficult to explain. It is best understood as taking a single argument, preprocessing it by a monadic function, and then passing both the preprocessed and unpreprocessed versions to a dyadic function. For example, (S prod log) multiplies a number by its logarithm:

$$(\mathbf{S}\ \text{prod log})\ p \Rightarrow \text{prod}\ p\ (\log p)$$

3.2.11 Psi Combinator: $\Psi \Rightarrow \lambda(f\,g\,x\,y)|f\,(g\,x)(g\,y)|$

Comments: This combinator preprocesses both arguments of a dyadic function by the same monadic function. Thus (Ψ sum log) is a function to add logarithmically:

$$(\Psi\ \text{sum log})\ a\ b \Rightarrow \text{sum}\ (\log a)\ (\log b)$$

Note that

$$\Psi\ f\ g \Leftrightarrow \Omega\ f\ g\ g \Leftrightarrow f\ :(g \cdot 1\text{st},\ g \cdot 2\text{nd}).$$

3.2.12 Paradozical Combinator: $\mathbf{Y} \Rightarrow \lambda f\ \{\ \lambda x\,|f\,(x\,x)|\ \lambda x\,|f\,(x\,x)|\ \}$

Comments: The Y combinator is difficult to explain. This definition of Y satisfies the functional equation:

$$\mathbf{Y}\ F \Rightarrow F(\mathbf{Y}\ F) \Rightarrow F(F(\mathbf{Y}\ F)) \Rightarrow \cdots$$

To see this observe

$$\mathbf{Y}\ F \Rightarrow \lambda x\,|F(x\,x)|\ \lambda x\,|F(x\,x)| \Rightarrow F\{\lambda x\,|F(x\,x)\ \lambda x\,|F(x\,x)|\} \Rightarrow \cdots$$

Thus it can be considered the *fixed-point finding operation.*

3.2.13 Recursive Definition: `rec F = D ' ⇒ 'F ⇒ Y λFD '

Comments: For example, a definition of the form

$$\text{rec fac} = \lambda n \begin{cases} n = 0 \rightarrow 1 \\ \text{else}\ n \times |\text{fac}(n-1)| \end{cases}$$

is to be replaced by

$$\text{fac} \Rightarrow \mathbf{Y}\ \lambda\text{fac}\left[\lambda n\begin{cases} n = 0 & \cdot\ 1 \\ \textbf{else}\ n & \times\ [\text{fac}\ (n-1)] \end{cases}\right]$$

The **Y** combinator in effect embeds the definition of fac in itself.

3.2.14 Seminormal Form: We say that a formula is in *seminormal form* if and only if the only reductions that can be performed on it and its descendants are of the form $\mathbf{Y}\ F \Rightarrow F(\ Y\ F)$.

Comments: Because of their use of the paradoxical combinator many useful formulas, for example the recursive definition of 'fac' exhibited above, do not have a normal form. Conceptually, however, this formula represents a value, *viz.*, the factorial function. If we view any application of **Y** as being a *potential* application, that is only performed when necessary to allow the reduction of non-**Y** applications, then we can see that a formula is in seminormal form when none of its *actual* applications can be reduced. That is, a formula is in seminormal form if either:

1. it is in normal form, or
2. it is not in normal form, but the only applications that can be reduced are **Y** applications, and performing these **Y** applications would not enable any non-**Y** applications.

Thus a formula in seminormal form can be thought of as *dormant*. It can be *activated* by using it in an application. For example, if $\lambda(xy)E$ is in normal form, then $\mathbf{Y}[\lambda(xy)E]$ is in seminormal form. This function can be activated by applying it to an argument a, since this application allows non-**Y** reductions to take place:

$$\{\mathbf{Y}[\lambda(xy)E]\}\ a \Rightarrow \lambda(xy)E\ \{\mathbf{Y}[\lambda(xy)E]\}\ a$$
$$\Rightarrow E\{x \leftarrow \{\mathbf{Y}[\lambda(xy)E]\},\ y \leftarrow a\}$$

We loosely use the term *defined* to mean that a formula has either a normal or seminormal form, and only discriminate between the two when necessary.

3.3 Conditionals

3.3.1 True Value: $\textbf{true} \Rightarrow \lambda(xy)x$

Comments: Note that

$$\textbf{true}\ a\ b \Rightarrow \lambda(xy)x\ a\ b \Rightarrow a$$

Thus **true** selects the first of its two arguments.

3.3.2 False Value: **false** $\Rightarrow \lambda(xy)y$

Comments: Note that

$$\textbf{false}\ a\ b \Rightarrow \lambda(xy)y\ a\ b \Rightarrow b$$

Thus **false** selects the second of its two arguments.

3.3.3 Definition of Boolean Value:

1. The normal form of '**true**' is a *Boolean value*.
2. The normal form of '**false**' is a *Boolean value*.
3. The only *Boolean values* are those described in (1) and (2) above.

3.3.4 Negation: not $\Rightarrow \lambda x(x\ \textbf{false true})$

Comments: For example,

not **true** $\Rightarrow \lambda x(x\ \textbf{false true})\ \textbf{true} \Rightarrow \textbf{true false true} \Rightarrow \lambda(xy)x\ \textbf{false true} \Rightarrow \textbf{false}$

3.3.5 Conjunction: and $\Rightarrow \lambda(xy)(x\ y\ \textbf{false})$

Comments: For example,

and **true false** $\Rightarrow$ **true false false** $\Rightarrow \lambda(xy)x$ **false false** $\Rightarrow$ **false**

3.3.6 Disjunction: or $\Rightarrow \lambda(xy)(x\ \textbf{true}\ y)$

Comments: For example,

or **false true** $\Rightarrow$ **false false true** $\Rightarrow$ **true**

3.3.7 Equivalence: equiv $\Rightarrow$ **S** x not y

Comments: For example,

equiv **true true** $\Rightarrow$ **S true** not **true** $\Rightarrow$ **true true** (not **true**) $\Rightarrow$ **true**

equiv **false true** $\Rightarrow$ **S false** not **true** $\Rightarrow$ **false true** (not **true**) $\Rightarrow$ not **true** $\Rightarrow$ **false**

3.3.8 Dyadic Conditional:

$$\begin{cases} P \rightarrow C \\ \textbf{else } A \end{cases} \quad \Rightarrow \quad P\ C\ A$$

Comments: This depends on the fact that if P is a Boolean value, then it will choose between C and A depending on whether it is **true** or **false**.

3.3.9 N-adic Conditional:

$$\begin{cases} P_1 \rightarrow C_1 \\ P_2 \rightarrow C_2 \\ \quad \vdots \\ P_n \rightarrow C_n \\ \textbf{else } A \end{cases} \quad \Rightarrow \quad P_1\ C_1\ (P_2\ C_2\ \cdots\ (P_n\ C_n\ A)\ \cdots\ )$$

3.4 Lists

3.4.1 First of Three: 1of3 $\Rightarrow \lambda(xyz)z$

Comments: Note that for any a, b, c:

$$\text{1of3}\ a\ b\ c \ \Rightarrow\ [\lambda(xyz)z]\ a\ b\ c \ \Rightarrow\ a$$

Notice that 1of3 selects the first of three values in much the same way that **true** selects the first of two values. In effect the formulas 1of3, 2of3 and 3of3 form the truth values of a three-valued logic.

3.4.2 Second of Three: 2of3 $\Rightarrow \lambda(xyz)y$

3.4.3 Third of Three: 3of3 $\Rightarrow \lambda(xyz)z$

3.4.4 List Value: list $\Rightarrow \lambda(nht)[\lambda s\,(s\ n\ h\ t)]$

Comments: This should be thought of as a triple (n, h, t), in which n is the *null-flag* and h and t are the head and tail of the list. The latter have significant values only if the null-flag is false. The parameter s can be thought of as a *potential selector*, which will select one of n, h or t. Notice that

$$\text{list } \textbf{false}\ A\ B \ \Rightarrow\ \lambda s\,(s\ \textbf{false}\ A\ B)$$

3.4.5 Null List: nil $\Rightarrow$ list **true** $\perp$ $\perp$

Comments: The first component of the triple, **true**, indicates that the list is null, and that the remaining two components are not used. The symbol '$\perp$' has no special significance, although it is intended to suggest an undefined value; it is merely a token like any other in the lambda calculus. Note that

$$\text{nil} \Rightarrow \text{list } \mathbf{true} \perp \perp \Rightarrow \lambda s\,(s\ \mathbf{true} \perp \perp)$$

which is the normal form for the null list.

3.4.6 List Construction: cons $\Rightarrow$ $\lambda(ht)$(list **false** h t)

Comments: For example,

$$\text{cons } A\ B \Rightarrow \text{list } \mathbf{false}\ A\ B \Rightarrow \lambda s\,(s\ \mathbf{false}\ A\ B)$$

That is, the result of 'cons A B' is a list triple, whose first element is **false** (meaning the list is nonnull), and whose second and third elements are A and B, the head and tail of the list.

3.4.7 Null Test: null $\Rightarrow$ $\lambda x\,(x$ 1of3)

Comments: This simply selects the first (null-flag) component of a list value:

$$\text{null nil} \Rightarrow \text{nil 1of3} \Rightarrow \lambda s\,(s\ \mathbf{true} \perp \perp)\ \text{1of3} \Rightarrow \text{1of3 } \mathbf{true} \perp \perp \Leftrightarrow \mathbf{true}$$

3.4.8 Head of List: hd $\Rightarrow$ $\lambda x\,(x$ 2of3)

Comments: This simply selects the second of the three components of a list value:

$$\text{hd(cons } A\ B) \Rightarrow (\text{cons } A\ B)\ \text{2of3} \Rightarrow \lambda s\,(s\ \mathbf{false}\ A\ B)\ \text{2of3} \Rightarrow \text{2of3 } \mathbf{false}\ A\ B \Rightarrow A$$

Note that hd nil $\Leftrightarrow$ $\perp$.

3.4.9 Tail of List: tl $\Rightarrow$ $\lambda x\,(x$ 3of3)

Comments: This simply selects the third of the three elements of a list value. We have that tl(cons A B) $\Leftrightarrow$ B. Similarly, it is easy to show that if L is a nonnull list then

$$\text{cons (hd } L)\ (\text{tl } L) \Leftrightarrow L$$

Simply observe that if L is a nonnull list then it has the form (list **false** A B). Note also that tl nil $\Leftrightarrow$ $\perp$.

3.4.10 List Notation:

$$<x_1, x_2, \ldots, x_n> \Rightarrow \text{cons } x_1 \text{ (cons } x_2 \cdots \text{ (cons } x_n \text{ nil) } \cdots \text{)}$$

More formally,

$< > \Rightarrow$ nil

$<x_1, x_2, \ldots, x_n> \Rightarrow \text{cons } x_1 <x_2, \ldots, x_n>$

Notice that by previously derived properties:

$\text{hd } <x_1, x_2, \ldots, x_n> \Rightarrow \text{hd(cons } x_1 <x_2, \ldots, x_n>) \Rightarrow x_1$

$\text{tl } <x_1, x_2, \ldots, x_n> \Rightarrow \text{tl(cons } x_1 <x_2, \ldots, x_n>) \Rightarrow <x_2, \ldots, x_n>$

3.4.11 Pair List: $\text{pair} \Rightarrow \lambda(xy)<x,y>$

Comments: The formula 'pair x y' constructs a two-element list $<x,y>$. Note that 'pair a' is a function that constructs a list whose first element is a:

$$(\text{pair } a)\ y \Rightarrow \text{pair } a\ y \Rightarrow <a,y>$$

3.4.12 Definition of List:

1. The normal form of 'nil' is a *list*.
2. If L is a list and x is any formula having a normal form, then the normal form of 'cons x L' is a *list*.
3. The only *lists* are those described in (1) and (2) above.

3.4.13 Example Normal Form:

For concreteness we exhibit the normal form of a particular list. Suppose '1' and 'a' are two tokens *(symbols in ν)*. Then we derive the normal form of the list '<1,a>' as follows:

$<1,a> \Rightarrow$ cons 1 (cons a nil)

$\Rightarrow$ (list **false** 1 (cons a nil))

$\Rightarrow$ (list **false** 1 (list **false** a nil))

$\Rightarrow$ (list **false** 1 (list **false** a (list **true** $\perp$ $\perp$)))

$\Rightarrow$ $\lambda s\,(s$ **false** 1 (list **false** a (list **true** $\perp$ $\perp$)))

$\Rightarrow$ $\lambda s\,(s$ **false** 1 $\lambda s\,(s$ **false** a (list **true** $\perp$ $\perp$)))

$\Rightarrow$ $\lambda s\,(s$ **false** 1 $\lambda s\,(s$ **false** a $\lambda s\,(s$ **true** $\perp$ $\perp$)))

$\Rightarrow$ $\lambda s\,(s\ \lambda(xy)y\ 1\ \lambda s\,(s\ \lambda(xy)y\ a\ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)))$

The final formula above is strictly speaking not in normal form, which would require replacing the suppressed parentheses. We omit this step as it would make the formula too unreadable.

3.4.14 Nonstandard Objects:

This representation permits several kinds of *nonstandard* lists. In particular, the structure (list **true** x y) behaves exactly like nil, the empty list, but contains two pieces of information, namely x and y. Also, although in the application (cons x y) the argument y is expected to be a list, this is not necessary. Whatever it is, the tl function will extract it.

4. Natural Numbers

In this chapter we define a representation for natural numbers and for the basic operations on natural numbers. Natural numbers are represented in unary notation as lists of identical tokens.

4.1 Primitive Ideas

4.1.1 Definition of Natural Number:

A *natural number* is a (possibly null) list, all of whose elements are nil.

4.1.2 Numeric Denotations:

$0 \Rightarrow$ < >

$1 \Rightarrow$ <nil>

$2 \Rightarrow$ <nil,nil>

$3 \Rightarrow$ <nil,nil,nil>

$4 \Rightarrow$ <nil,nil,nil,nil>

$\vdots$

$9 \Rightarrow$ <nil,nil,nil,nil,nil,nil,nil,nil,nil>

4.1.3 Example Normal Form:

For concreteness we exhibit the normal form of a particular natural number:

$0 \Rightarrow$ nil

$\Rightarrow$ (list **true** $\perp$ $\perp$)

$\Rightarrow$ $\lambda s\,(s$ **true** $\perp$ $\perp)$

$\Rightarrow$ $\lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)$

4.1.4 Nonstandard Objects:

There are no nonstandard natural numbers, since they are defined to be lists of nils. However, it will be seen that the arithmetic operations and relations depend only on the lengths of these lists, so that lists of things other than nils can be considered nonstandard natural numbers. For example, the normal forms of <$\perp$>, <**true**>, <**false**>, <<nil>>, etc. could be considered alternate, nonstandard representations of

1.

4.1.5 Zero Predicate: $\text{zero?}_N \Rightarrow \text{null}$

Comments: For example,

$\text{zero?}_N\ 0 \Rightarrow \text{null} <> \Rightarrow \textbf{true}$

$\text{zero?}_N\ 1 \Rightarrow \text{null} <\text{nil}> \Rightarrow \textbf{false}$

4.1.6 Successor: $\text{succ}_N \Rightarrow \text{cons nil}$

Comments: Since the natural number n is represented by a list of n nils, the successor function simply adds another nil to the list. For example,

$$\text{succ}_N\ 3 \Rightarrow \text{cons nil} <\text{nil,nil,nil}> \Rightarrow <\text{nil,nil,nil,nil}> \Leftrightarrow 4$$

4.1.7 Predecessor: $\text{pred}_N \Rightarrow \text{tl}$

Comments: For example,

$$\text{pred}_N\ 4 \Rightarrow \text{tl} <\text{nil,nil,nil,nil}> \Rightarrow <\text{nil,nil,nil}> \Leftrightarrow 3$$

4.2 Arithmetic

4.2.1 Repeated Composition:

$$\textbf{rec rpt} = \lambda(f\,n\,a)\begin{cases}\text{zero?}_N\ n & \rightarrow\ a \\ \textbf{else}\ f\ [\text{rpt}\ f\ (\text{pred}_N\ n)\ a\,]\end{cases}$$

Comments: Notice that:

$\text{rpt}\ f\ 0\ x \Rightarrow x$

$\text{rpt}\ f\ 1\ x \Rightarrow f x$

$\text{rpt}\ f\ 2\ x \Rightarrow f\,(f x)$

$\text{rpt}\ f\ 3\ x \Rightarrow f\,(f\,(f x))$

$\vdots$

$\text{rpt}\ f\ n\ x \Rightarrow f^n x$

In general, rpt f n is the n-fold composition of f with itself.

4.2.2 Sum: $\text{sum}_N \Rightarrow \text{rpt succ}_N$

Comments: Thus, the sum of m and n is the mth successor of n:

$$\text{sum}_N\ m\ n \Rightarrow \text{rpt succ}_N\ m\ n \Rightarrow \text{succ}_N{}^m\, n \Rightarrow m+n$$

4.2.3 Difference: $\text{dif}_N \Rightarrow \text{conv}(\text{rpt pred}_N)$

Comments: The effect of this definition is that the difference of m and n is the nth predecessor of m:

$$\text{dif}_N\ m\ n \Rightarrow \text{conv}(\text{rpt pred}_N)\ m\ n \Rightarrow \text{rpt pred}_N\ n\ m \Rightarrow \text{pred}_N{}^n\, m \Rightarrow m-n$$

4.2.4 Product: $\text{prod}_N \Rightarrow \lambda(mn)[\text{rpt}\ (\text{sum}_N\ m)\ n\ 0]$

Comments: Notice that $(\text{sum}_N\ m)$ is the function that adds m. Hence the product of m and n is defined to be the result of adding m to zero n times:

$$\text{prod}_N\ m\ n \Rightarrow \text{rpt}\ (\text{sum}_N\ m)\ n\ 0 \Rightarrow (\text{sum}_N\ m)^n 0 \Rightarrow m+m+\cdots+m+0 \Rightarrow mn$$

4.3 Predicates and Relations

4.3.1 Less-Than Relation:

$$\textbf{rec}\ \ \text{lt}_N \ = \ \lambda(mn)\begin{cases}\text{zero?}_N\ n \ \rightarrow\ \textbf{false}\\ \text{zero?}_N\ m \ \rightarrow\ \textbf{true}\\ \textbf{else}\ \text{lt}_N\ (\text{pred}_N\ m)\ (\text{pred}_N\ n)\end{cases}$$

Comments: This definition is based on the fact that

$$m < n \iff m-1 < n-1$$

Thus, $\text{lt}_N\ m\ n$ operates recursively, subtracting one from each of m and n until one or the other reaches 0, at which point the truth value of the relation is known:

$\text{lt}_N\ m\ 0 \Rightarrow$ **false**

$\text{lt}_N\ 0\ n \Rightarrow$ **true**, if n nonzero

$\text{lt}_N\ m\ n \Rightarrow \text{lt}_N\ (m-1)\ (n-1)$, if m, n nonzero

4.3.2 Greater-Than Relation: $\text{gt}_N \Rightarrow \text{conv lt}_N$

Comments: Observe that

$$gt_N\ m\ n \Rightarrow \text{conv}\ lt_N\ m\ n \Rightarrow lt_N\ n\ m$$

4.3.3 Inequality Relation: $ne_N \Rightarrow or:(lt_N, gt_N)$

Comments: Two natural numbers are unequal if one is either greater or less than the other:

$$ne_N\ m\ n \Rightarrow or:(lt_N, gt_N)\ m\ n \Rightarrow or(lt_N\ m\ n)\ (gt_N\ m\ n)$$

4.3.4 Equality Relation: $eq_N \Rightarrow not:ne_N$

Comments: Two natural numbers are equal if they are not unequal:

$$eq_N\ m\ n \Rightarrow not:ne_N\ m\ n \Rightarrow not(ne_N\ m\ n)$$

4.3.5 Less-or-Equal Relation: $le_N \Rightarrow not:gt_N$

Comments: One number is less than or equal to another if it is not greater than the other:

$$le_N\ m\ n \Rightarrow not:gt_N\ m\ n \Rightarrow not(gt_N\ m\ n)$$

4.3.6 Greater-or-Equal Relation: $ge_N \Rightarrow not:lt_N$

4.4 Miscellaneous Operations

4.4.1 Quotient: $\text{rec}\ quo_N = \lambda(mn)\begin{cases} lt_N\ m\ n \rightarrow 0 \\ \text{else}\ succ_N:quo_N\ (dif_N\ m\ n)\ n \end{cases}$

Comments: The definition of quotient is based on the following identity:

$$m \div n = 1 + (m-n) \div n$$

Note that if m is greater or equal to n then

$$quo_N\ m\ N \Rightarrow succ_N:quo_N\ (dif_N\ m\ n)\ n \Rightarrow succ_N[quo_N\ dif_N\ m\ n)\ n]$$

4.4.2 Factorial: $\text{rec fac} = \lambda n\begin{cases} zero?_N\ n \rightarrow 1 \\ \text{else}\ prod_N\{n, [fac\ (pred_N\ n)]\} \end{cases}$

4.4.3 Definition of Sequence:

A *sequence* is any formula N such that for each nonzero natural number k the formula 'Nk' is defined.

If C is any class of normal forms and N is a sequence, then N is called a *sequence of Cs* if the normal form of each formula Nk is in C.

4.4.4 Subscript Notation:

If N is a sequence we usually write N_k for Nk:

$$N_k \Rightarrow Nk$$

5. Integers

In this chapter we define integers as pairs comprising a sign (represented by a Boolean value) and a magnitude (represented by a natural number).

5.1 Primitive Ideas

5.1.1 Definition of Integer:

An *integer* is a pair (two element list), whose first element is a Boolean value and whose second element is a natural number.

5.1.2 Attach Plus-sign: $\text{plus}_Z \Rightarrow$ pair **false**

Comments: This operation converts a natural number into a plus-signed integer by attaching a plus-flag (false value) to it. For example,

$$\text{plus}_Z\ 3 \Rightarrow \text{pair}\ \textbf{false}\ 3 \Rightarrow \langle \textbf{false}, 3 \rangle$$

5.1.3 Attach Minus-sign: $\text{minus}_Z \Rightarrow$ pair **true**

Comments: This operation converts a natural number into a minus-signed integer.

5.1.4 Denotations:

$+0 \Rightarrow \text{plus}_Z\ 0$

$+1 \Rightarrow \text{plus}_Z\ 1$

$+2 \Rightarrow \text{plus}_Z\ 2$

$-1 \Rightarrow \text{minus}_Z\ 1$

5.1.5 Example Normal Form:

For concreteness we exhibit the normal form of '+0':

$$\begin{aligned} +0 &\Rightarrow \text{plus}_Z\ 0 \\ &\Rightarrow \langle \textbf{false}, 0 \rangle \\ &\Rightarrow \langle \lambda(xy)y\ ,\ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp) \rangle \\ &\Rightarrow \lambda s\,(s\ \lambda(xy)y\ \ \lambda(xy)y\ \ \lambda s\,(s\ \lambda(xy)y\ \ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)\ \lambda s\,(s\ \lambda(xy)x\ \ \ \perp))) \end{aligned}$$

5.1.6 Nonstandard Objects:

This representation permits one nonstandard object, -0. This is because +0 and -0 have different representations, <**false**,0> and <**true**,0>, although the arithmetic operations and relations treat them as equivalent.

5.1.7 Extract Magnitude: $mag_Z \Rightarrow hd \circ tl$

Comments: This operation converts an integer into a natural number by discarding its sign: For example,

$$mag_Z\ +1 \Rightarrow mag_Z\ \langle \textbf{false},1\rangle \Rightarrow hd \circ tl\ \langle \textbf{false},1\rangle \Rightarrow hd(tl\ \langle \textbf{false},1\rangle) \Rightarrow hd\ \langle 1\rangle \Rightarrow 1$$

$$mag_Z\ -1 \Rightarrow mag_Z\ \langle \textbf{true},1\rangle \Rightarrow hd \circ tl\ \langle \textbf{true},1\rangle \Rightarrow hd(tl\ \langle \textbf{true},1\rangle) \Rightarrow hd\ \langle 1\rangle \Rightarrow 1$$

5.2 Primitive Predicates

5.2.1 Minus-sign Test: $minus?_Z \Rightarrow hd$

Comments: This operation tests if an integer bears a minus sign:

$$minus?_Z\ +1 \Rightarrow minus?_Z\ \langle \textbf{false},1\rangle \Rightarrow hd\ \langle \textbf{false},1\rangle \Rightarrow \textbf{false}$$

Notice that a number bearing a minus sign is not the same thing as it being negative, since -0 bears a minus sign. True positive and negative tests are defined later.

5.2.2 Plus-sign Test: $plus?_Z \Rightarrow not \circ minus?_Z$

Comments: For example,

$$plus?_Z\ +1 \Rightarrow not \circ minus?_Z\ +1 \Rightarrow not(minus?_Z\ +1) \Rightarrow not\ \textbf{false} \Rightarrow \textbf{true}$$

Again, bearing a plus sign is not the same as being positive, since +0 bears a plus sign.

5.2.3 Zero Test: $zero?_Z \Rightarrow zero?_N \circ mag_Z$

Comments: For example,

$$zero?_Z\ +1 \Rightarrow zero?_N \circ mag_Z\ +1 \Rightarrow zero?_N(mag_Z\ +1) \Rightarrow zero?_N\ 1 \Rightarrow \textbf{false}$$

5.3 Primitive Operations

5.3.1 Absolute Value: $\text{abs}_Z \Rightarrow \text{plus}_Z \circ \text{mag}_Z$

Comments: For example,

$$\text{abs}_Z\ -1 \Rightarrow \text{plus}_Z\,(\text{mag}_Z\ -1) \Rightarrow \text{plus}_Z\ 1 \Leftrightarrow +1$$

5.3.2 Negate: $\text{neg}_Z \Rightarrow \text{pair} \circ (\text{plus?}_Z, \text{mag}_Z)$

Comments: This operation negates an integer. For example,

$$\text{neg}_Z\ -1 \Rightarrow \text{pair} \circ (\text{plus?}_Z, \text{mag}_Z)\ -1$$

$$\Rightarrow \text{pair}\ (\text{plus?}_Z\ -1)\ (\text{mag}_Z\ -1) \Rightarrow \text{pair}\ \textbf{false}\ 1 \Leftrightarrow +1$$

5.3.3 Successor: $\text{succ}_Z \Rightarrow \lambda n \begin{cases} \text{zero?}_Z\ n & \rightarrow +1 \\ \text{plus?}_Z\ n & \rightarrow \text{plus}_Z \circ \text{succ}_N \circ \text{mag}_Z\ n \\ \text{minus?}_Z\ n & \rightarrow \text{minus}_Z \circ \text{pred}_N \circ \text{mag}_Z\ n \end{cases}$

Comments: This operation computes the successor of an integer. The main complication is to avoid treating -0 and +0 as different integers. The cases are:

$$\text{succ}_Z\ \pm 0 \Rightarrow +1$$

$$\text{succ}_Z\ +n \Rightarrow \text{plus}_Z\,[\text{succ}_N\,(\text{mag}_Z\ +n)] \Rightarrow +(n+1)$$

$$\text{succ}_Z\ -n \Rightarrow \text{minus}_Z\,[\text{pred}_N\,(\text{mag}_Z\ -n)] \Rightarrow -(n-1)$$

5.3.4 Predecessor: $\text{pred}_Z \Rightarrow \text{neg}_Z \circ \text{succ}_Z \circ \text{neg}_Z$

Comments: Notice that

$$\text{pred}_Z\ n \Rightarrow \text{neg}_Z \circ \text{succ}_Z \circ \text{neg}_Z\ n \Rightarrow \text{neg}_Z\,[\text{succ}_Z\,(\text{neg}_Z\ n)]$$

$$= -[1 + (-n)] = -[1-n] = n-1$$

5.3.5 Sum: $\textbf{rec}\ \text{sum}_Z = \lambda(mn) \begin{cases} \text{zero?}_Z\ n & \rightarrow m \\ \text{minus?}_Z\ n & \rightarrow \text{pred}_Z\,[\text{sum}_Z\ m\ (\text{succ}_Z\ n)] \\ \text{plus?}_Z\ n & \rightarrow \text{succ}_Z\,[\text{sum}_Z\ m\ (\text{pred}_Z\ n)] \end{cases}$

Comments: The sum of integers is defined recursively by the following cases:

$m + 0 = m$

$m + n = [m + (n+1)] - 1$, if $n < 0$

$m + n = [m + (n-1)] + 1$, if $n > 0$

5.3.6 Difference: $\text{dif}_Z \Rightarrow \text{sum}_Z : (\text{1st}, \text{neg}_Z \circ \text{2nd})$

Comments: Note that

$$\text{dif}_Z\ m\ n \Rightarrow \text{sum}_Z : (\text{1st}, \text{neg}_Z \circ \text{2nd})\ m\ n \Rightarrow \text{sum}_Z\ m\ (\text{neg}_Z\ n)$$

$$\Rightarrow \text{sum}_Z\ m\ (\text{neg}_Z\ n) = m + (-n) = m - n$$

5.4 Predicates and Relations

5.4.1 Equality: $\text{eq}_Z \Rightarrow \text{zero?}_Z : \text{dif}_Z$

Comments: This depends on the fact that $m = n$ if and only if $m - n$ is zero:

$$\text{eq}_Z\ m\ n \Rightarrow \text{zero?}_Z : \text{dif}_Z\ m\ n \Rightarrow \text{zero?}_Z\ (\text{dif}_Z\ m\ n)$$

5.4.2 Negative Predicate: $\text{neg?}_Z \Rightarrow \text{and} \circ (\text{minus?}_Z, \text{not} \circ \text{zero?}_Z)$

Comments: This is a true negative test (in contrast to minus?_Z), and depends on the fact that $m < 0$ if and only if m bears a minus sign and is nonzero:

$$\text{neg?}_Z\ m \Rightarrow \text{and} \circ (\text{minus?}_Z, \text{not} \circ \text{zero?}_Z)\ m \Rightarrow \text{and}\ (\text{minus?}_Z\ m)\ [\text{not}(\text{zero?}_Z\ m)]$$

5.4.3 Positive Predicate: $\text{pos?}_Z \Rightarrow \text{and}\textit{cric}\ (\text{plus?}_Z, \text{not} \circ \text{zero?}_Z)$

Comments: An integer is positive if it is nonzero and bears a plus sign:

$$\text{pos?}_Z\ m \Rightarrow \text{and} \circ (\text{plus?}_Z, \text{not} \circ \text{zero?}_Z)\ m \Rightarrow \text{and}\ (\text{plus?}_Z\ m)\ [\text{not}(\text{zero?}_Z\ m)]$$

5.4.4 Less-Than Relation: $\text{lt}_Z \Rightarrow \text{neg?}_Z : \text{dif}_Z$

Comments: We make use of the fact that $m < n$ if and only if $m - n$ is negative:

$$\text{lt}_Z\ m\ n \Rightarrow \text{neg?}_Z : \text{dif}_Z\ m\ n \Rightarrow \text{neg?}_Z\ (\text{dif}_Z\ m\ n)$$

5.4.5 Greater-Than Relation: $\text{gt}_Z \Rightarrow \text{pos?}_Z : \text{dif}_Z$

Comments: Note

$$gt_Z\ m\ n \Rightarrow pos?_Z{:}dif_Z\ m\ n \Rightarrow pos?_Z(dif_Z\ m\ n)$$

5.4.6 Greater-or-Equal Relation: $ge_Z \Rightarrow not{:}lt_Z$

Comments: Here we make use of the fact that $m \geq n$ is true if and only if $m < n$ is false:

$$ge_Z\ m\ n \Rightarrow not{:}lt_Z\ m\ n \Rightarrow not(lt_Z\ m\ n)$$

5.4.7 Less-or-Equal Relation: $le_Z \Rightarrow not{:}gt_Z$

5.4.8 Maximum: $max_Z \Rightarrow \lambda(zy)\begin{cases}(ge_Z\ z\ y) \rightarrow z\\ \textbf{else}\ y\end{cases}$

5.5 Multiplicative Operations

5.5.1 Product:

$$prod_Z \Rightarrow \lambda(mn)\begin{cases}\text{equiv}\ (neg_Z\ m)\ (neg_Z\ n) \rightarrow plus_Z{:}prod_N\ (mag_Z\ m)\ (mag_Z\ n)\\ \textbf{else}\ minus_Z{:}prod_N\ (mag_Z\ m)\ (mag_Z\ n)\end{cases}$$

Comments: This definition operates by attaching the appropriate sign to the result of a natural number multiplication of the magnitudes of the factors. The only complication is to take care of the sign of the result. In this connection note that

$$\text{equiv}\ (neg_Z\ m)\ (neg_Z\ n)$$

is true if and only if the signs of m and n are the same.

5.5.2 Quotient:

$$quo_Z \Rightarrow \lambda(mn)\begin{cases}\text{equiv}\ (neg_Z\ m)\ (neg_Z\ n) \rightarrow plus_Z{:}quo_N\ (mag_Z\ m)\ (mag_Z\ n)\\ \textbf{else}\ minus_Z{:}quo_N\ (mag_Z\ m)\ (mag_Z\ n)\end{cases}$$

Comments: Like the product operation, quotient operates by attaching the appropriate sign to the result of the corresponding natural number operation.

6. Rational Numbers

In this chapter rational numbers and their operations are defined, based on the representation of a rational number as a pair of integers.

6.1 Primitive Ideas

6.1.1 Definition of Rational Number:

A *rational number* is a pair (two element list), each of whose elements is an integer.

6.1.2 Construction: $\text{rat}_Q \Rightarrow \text{pair}$

Comments: Note that

$$\text{rat}_Q\ m\ n \Rightarrow \text{pair}\ m\ n \Rightarrow \langle m,n \rangle$$

6.1.3 Notation: $m / n \Rightarrow \langle m,n \rangle$

6.1.4 Denotations:

$m / 1 \Rightarrow m / +1$

$0 / 1 \Rightarrow +0 / 1$

$1 / 1 \Rightarrow +1 / 1$

$2 / 1 \Rightarrow +2 / 1$

$1 / 2 \Rightarrow +1 / +2$

6.1.5 Example Normal Form:

For concreteness we exhibit the normal form of 0/0:

$$0/0 \Rightarrow \langle +0,+0 \rangle$$

$$\Rightarrow \lambda s\,(s\ \textbf{false}\ +0\ \lambda s\,(s\ \textbf{false}\ -0\ \lambda s\,(s\ \textbf{true}\ \perp\ \perp)))$$

$$\Rightarrow \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)$$
$$\lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)))$$
$$\lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)$$
$$\lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)))\ \lambda s\,(s\ \lambda(xy)x\ \perp\ \perp)))$$

The normal form of 1/2 is shown in Figure 1.

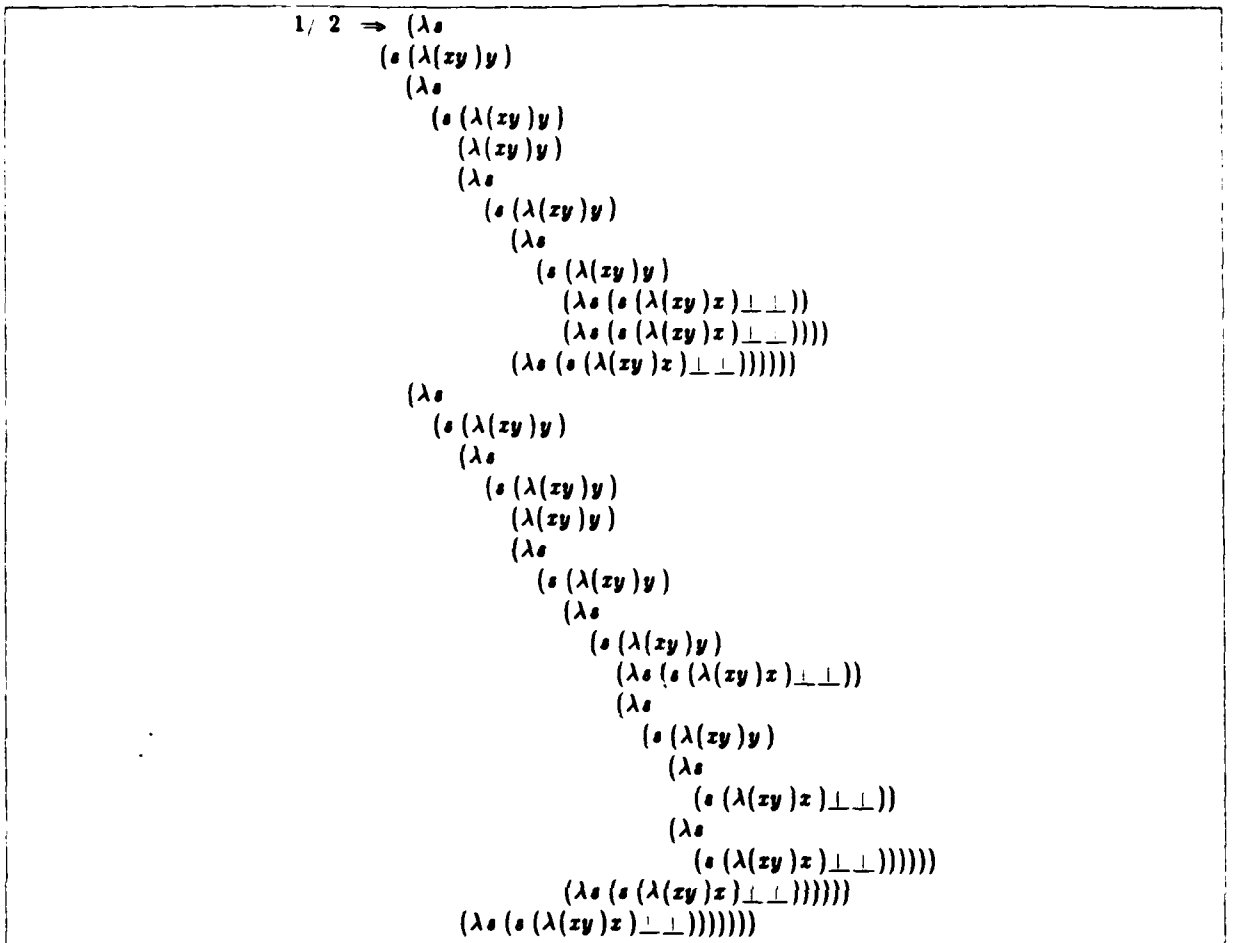


Figure 1. Normal Form of 1/2

6.1.6 Nonstandard Objects:

There are many equivalent representations of the same rational number; alternately, these may be considered different (nonstandard) rational numbers that behave the same under the arithmetic operations. For example, +1/+2, +2/+4, -1/-2, -2/-4 etc. all behave the same. Similarly, +0/+1, +0/-1, +0/+2, -0/-1, etc. all behave the same. "Infinite" rational numbers, such as +1/+0, -1/-0, etc. are also permitted by the representation, and can be considered nonstandard objects. Operations on nonstandard objects are discussed further at the end of this section.

6.1.7 Extract Numerator: $\text{num}_Q \Rightarrow \text{hd}$

Comments: Note that

$$\text{num}_Q(m\,/\,n) \Rightarrow \text{hd}(m\,/\,n) \Rightarrow \text{hd}\,\langle m,n\rangle \Rightarrow m$$

6.1.8 Extract Denominator: $\text{den}_Q \Rightarrow \text{hd} \circ \text{tl}$

Comments: Note that

$$\text{den}_Q\,(m\,/\,n) \Rightarrow \text{hd} \circ \text{tl}\,(m\,/\,n) \Rightarrow \text{hd} \circ \text{tl} < m, n > \Rightarrow n$$

6.2 Arithmetic Operations

6.2.1 Sum:

$$\text{sum}_Q \Rightarrow \text{rat}_Q : [\text{sum}_Z : (\Omega\ \text{prod}_Z\ \text{num}_Q\ \text{den}_Q, \Omega\ \text{prod}_Z\ \text{den}_Q\ \text{num}_Q),$$
$$\Omega\ \text{prod}_Z\ \text{den}_Q\ \text{den}_Q]$$

Comments: Alternately, replacing Ω with a dyadic composition we have

$$\text{sum}_Q \Rightarrow \text{rat}_Q : \{\text{sum}_Z : [\text{prod}_Z : (\text{num}_Q \circ 1\text{st}, \text{den}_Q \circ 2\text{nd}), \text{prod}_Z : (\text{den}_Q \circ 1\text{st}, \text{num}_Q \circ 2\text{nd})],$$
$$\text{prod}_Z : (\text{den}_Q \circ 1\text{st}, \text{den}_Q \circ 2\text{nd})\}$$

This definition makes use of the equation

$$m\,/\,n + m'\,/\,n' = (mn' + nm')\,/\,nn'$$

To see this note that

$$\text{sum}_Q\ (m\,/\,n)\ (m'\,/\,n')$$
$$\Rightarrow \text{rat}_Q : [\text{sum}_Z : (\Omega\ \text{prod}_Z\ \text{num}_Q\ \text{den}_Q, \Omega\ \text{prod}_Z\ \text{den}_Q\ \text{num}_Q),$$
$$\Omega\ \text{prod}_Z\ \text{den}_Q\ \text{den}_Q]\ (m\,/\,n)\ (m'\,/\,n')$$
$$\Rightarrow \text{rat}_Q\ [\text{sum}_Z : (\Omega\ \text{prod}_Z\ \text{num}_Q\ \text{den}_Q, \Omega\ \text{prod}_Z\ \text{den}_Q\ \text{num}_Q)\ (m\,/\,n)\ (m'\,/\,n')]$$
$$[(\Omega\ \text{prod}_Z\ \text{den}_Q\ \text{den}_Q)\ (m\,/\,n)\ (m'\,/\,n')]$$

Now note that

$$(\Omega\ \text{prod}_Z\ \text{den}_Q\ \text{den}_Q)\ (m\,/\,n)\ (m'\,/\,n')$$
$$\Rightarrow \text{prod}_Z\ [\text{den}_Q\,(m\,/\,n)]\ \text{den}_Q\,(m'\,/\,n')] \Rightarrow \text{prod}_Z\ n\ n' = nn'$$

Also note that

sum_Z :(Ω prod_Z num_Q den_Q , Ω prod_Z den_Q num_Q) (m / n) (m' / n')

$\Rightarrow$ sum_Z [Ω prod_Z num_Q den_Q (m / n) (m' / n')] [Ω prod_Z den_Q num_Q (m / n) (m' / n')]

$\Rightarrow$ sum_Z {prod_Z [num_Q (m / n)] [den_Q (m' / n')]}

{prod_Z [den_Q (m / n)] [num_Q (m' / n')]}

$\Rightarrow$ sum_Z (prod_Z m n') (prod_Z n m')

$=$ $mn' + nm'$

6.2.2 Absolute Value: abs_Q $\Rightarrow$ rat_Q ∘(abs_Z ∘num_Q , abs_Z ∘den_Q)

Comments: This definition is based on

$$| m / n | = | m | / | n |$$

To see this note

abs_Q (m / n) $\Rightarrow$ rat_Q ∘(abs_Z ∘num_Q , abs_Z ∘den_Q) (m / n)

$\Rightarrow$ rat_Q [(abs_Z ∘num_Q) (m / n)] [(abs_Z ∘den_Q) (m / n)]

$\Rightarrow$ rat_Q (abs_Z m) (abs_Z n)

6.2.3 Negation: neg_Q $\Rightarrow$ rat_Q ∘(neg_Z ∘num_Q , den_Q)

Comments: This is based on $-(m / n) = (-m) / n$. To see this note

neg_Q (m / n) $\Rightarrow$ rat_Q ∘(neg_Z ∘num_Q , den_Q) (m / n)

$\Rightarrow$ rat_Q [(neg_Z ∘num_Q) (m / n)] [den_Q (m / n)]

$\Rightarrow$ rat_Q (neg_Z m) n

6.2.4 Difference: dif_Q $\Rightarrow$ sum_Q :(1st, neg_Q ∘2nd)

Comments: Note that

dif_Q r s $\Rightarrow$ sum_Q :(1st,neg_Q ∘2nd) r s $\Rightarrow$ $\Rightarrow$ sum_Q r (neg_Q s)

6.2.5 Product: prod_Q $\Rightarrow$ rat_Q :(Ψ prod_Z num_Q , Ψ prod_Z den_Q)

Comments: This is based on the identity

$$(m / n) (m' / n') = (mm') / (nn')$$

Note that

$$\text{prod}_Q\ (m\ /\ n)\ (m'\ /\ n') \Rightarrow \text{rat}_Q\ :(\Psi\ \text{prod}_Z\ \text{num}_Q\ ,\ \Psi\ \text{prod}_Z\ \text{den}_Q)\ (m\ /\ n)\ (m'\ /\ n')$$

$$\Rightarrow \text{rat}_Q\ [(\Psi\ \text{prod}_Z\ \text{num}_Q)\ (m\ /\ n)\ (m'\ /\ n')]\ [(\Psi\ \text{prod}_Z\ \text{den}_Q)\ (m\ /\ n)\ (m'\ /\ n')]$$

Now note that

$$(\Psi\ \text{prod}_Z\ \text{num}_Q)\ (m\ /\ n)\ (m'\ /\ n') \Rightarrow \text{prod}_Z\ [\text{num}_Q\ (m\ /\ n)]\ [\text{num}_Q\ (m'\ /\ n')]$$

$$\Rightarrow \text{prod}_Z\ m\ m'$$

Similarly,

$$(\Psi\ \text{prod}_Z\ \text{den}_Q)\ (m\ /\ n)\ (m'\ /\ n') \Rightarrow \text{prod}_Z\ n\ n'$$

6.2.6 Reciprocal:

$$\text{recip}_Q \Rightarrow \text{rat}_Q\ \circ(\text{den}_Q\ ,\text{num}_Q)$$

Comments: Note that

$$\text{recip}_Q\ (m\ /\ n) \Rightarrow \text{rat}_Q\ \circ(\text{den}_Q\ ,\text{num}_Q)\ (m\ /\ n)$$

$$\Rightarrow \text{rat}_Q\ [\text{den}_Q\ (m\ /\ n)]\ [\text{num}_Q\ (m\ /\ n)] \Rightarrow \text{rat}_Q\ n\ m \Leftrightarrow n\ /\ m$$

Notice that the reciprocal of zero is defined (has a normal form) since $\text{recip}_Q\ 0/\ 1 \Leftrightarrow +1/\ +0$, although this is a nonstandard object.

6.2.7 Quotient: $\text{quo}_Q \Rightarrow \text{prod}_Q\ :(1\text{st},\ \text{recip}_Q\ \circ 2\text{nd})$

Comments: Note that

$$\text{quo}_Q\ r\ s \Rightarrow \text{prod}_Q\ :(1\text{st},\ \text{recip}_Q\ \circ 2\text{nd})\ r\ s \Rightarrow \text{prod}_Q\ r\ (\text{recip}_Q\ s)$$

6.2.8 Floor: $\text{floor}_Q \Rightarrow \text{quo}_Z\ \circ(\text{num}_Q\ ,\text{den}_Q)$

Comments: The floor of a rational number is the greatest integer less than that rational number. Thus the floor of $m\ /\ n$ is the integer part of the quotient of m and n:

$$\text{floor}_Q\ (m\ /\ n) \Rightarrow \text{quo}_Z\ \circ(\text{num}_Q\ ,\text{den}_Q)\ (m\ /\ n)$$

$$\Rightarrow \text{quo}_Z\ [\text{num}_Q\ (m\ /\ n)]\ [\text{den}_Q\ (m\ /\ n)] \Rightarrow \text{quo}_Z\ m\ n$$

6.2.9 Ceiling: $\text{ceil}_Q \Rightarrow \text{floor}_Q \circ [\text{sum}_Q\ (1/\ 2)]$

Comments: The ceiling of a rational number is the least integer greater than or equal to that rational number. To compute this we use the relation

$$\text{ceil}_Q\ r\ =\ \text{floor}_Q\,(r\ +\ \tfrac{1}{2})$$

Note that

$$\text{ceil}_Q\ r\ \Rightarrow\ \text{floor}_Q \circ [\text{sum}_Q\ (1/\ 2)]\ r\ \Rightarrow\ \text{floor}_Q\,[\text{sum}_Q\ (1/\ 2)\ r\,]$$

6.3 Predicates and Relations

6.3.1 Nonnegative: $\text{nonneg?}_Q \Rightarrow \text{equiv} \circ (\text{neg?}_Z \circ \text{num}_Q\,,\ \text{neg?}_Z \circ \text{den}_Q\,)$

Comments: A rational number is nonnegative if both its numerator and denominator have the same sign. That is $m\ /\ n\ \geqslant 0$ if and only if

$$\text{equiv}\ (\text{neg?}_Z\ m)\ (\text{neg?}_Z\ n\,)\ \Leftrightarrow\ \textbf{true}$$

Note that

$$\begin{aligned}\text{nonneg?}_Q\ (m\ /\ n\,)\ &\Rightarrow\ \text{equiv} \circ (\text{neg?}_Z \circ \text{num}_Q\,,\ \text{neg?}_Z \circ \text{den}_Q\,)\ (m\ /\ n\,)\\ &\Rightarrow\ \text{equiv}\ [(\text{neg?}_Z \circ \text{num}_Q\,)\ (m\ /\ n\,)]\ [(\text{neg?}_Z \circ \text{den}_Q\,)\ (m\ /\ n\,)]\\ &\Rightarrow\ \text{equiv}\ (\text{neg?}_Z\ m)\ (\text{neg?}_Z\ n\,)\end{aligned}$$

6.3.2 Negative: $\text{neg?}_Q \Rightarrow \text{not} \circ \text{nonneg?}_Q$

Comments: A rational is negative if it is not nonnegative.

6.3.3 Less-Than: $\text{lt}_Q \Rightarrow \text{neg?}_Q : \text{dif}_Q$

Comments: A rational r is less than a rational s if and only if $r - s$ is negative:

$$\text{lt}_Q\ r\ s\ \Rightarrow\ \text{neg?}_Q : \text{dif}_Q\ r\ s\ \Rightarrow\ \text{neg?}_Q\,(\text{dif}_Q\ r\ s\,)$$

6.3.4 Greater-Than: $\text{gt}_Q \Rightarrow \text{conv lt}_Q$

Comments: Note that $\text{gt}_Q\ r\ s\ \Rightarrow\ \text{conv gt}_Q\ r\ s\ \Rightarrow\ \text{gt}_Q\ s\ r$

6.3.5 Less or Equal: $le_Q \Rightarrow not{:}gt_Q$

6.3.6 Greater or Equal: $ge_Q \Rightarrow not{:}lt_Q$

6.3.7 Equality: $eq_Q \Rightarrow eq_Z{:}(\Omega\ prod_Z\ num_Q\ den_Q, \Omega\ prod_Z\ den_Q\ num_Q)$

Comments: Alternately, we can replace the Ωs with dyadic compositions:

$$eq_Q \Rightarrow eq_Z{:}[prod_Z{:}(num_Q \circ 1st, den_Q \circ 2nd),$$
$$prod_Z{:}(den_Q \circ 1st, num_Q \circ 2nd)]$$

Two rationals m / n and m' / n' are equal if and only if $mn' = m'n$. Note

$$eq_Q\ (m / n)\ (m' / n')$$
$$\Rightarrow eq_Z{:}(\Omega\ prod_Z\ num_Q\ den_Q, \Omega\ prod_Z\ den_Q\ num_Q)\ (m / n)\ (m' / n')$$
$$\Rightarrow eq_Z\ [(\Omega\ prod_Z\ num_Q\ den_Q)\ (m / n)\ (m' / n')]\ [(\Omega\ prod_Z\ num_Q\ den_Q)\ (m / n)\ (m' / n')]$$
$$\Rightarrow eq_Z\ \{prod_Z\ [num_Q\ (m / n)]\ [den_Q\ (m' / n')]\}\ \{prod_Z\ [den_Q\ (m / n)]\ [num_Q\ (m' / n')]\}$$
$$\Rightarrow eq_Z\ (prod_Z\ m\ n')\ (prod_Z\ n\ m')$$

6.3.8 Maximum: $max_Q \Rightarrow \lambda(xy)\begin{cases}(ge_Q\ x\ y) \rightarrow x\\ \text{else } y\end{cases}$

6.4 Operations on Nonstandard Rational Numbers

6.4.1 Infinite and Indefinite Rational Numbers:

$$\infty \Rightarrow +1/+0$$
$$-\infty \Rightarrow -1/+0$$
$$\chi \Rightarrow +0/+0$$

Comments: The chosen representation for rational numbers permits the nonstandard rationals 1/0, -1/0 and 0/0. We show later that other nonstandard rationals, such as 2/0 and -3/0, are equal to one of the three defined above.

In this section we explore the extent to which these nonstandard numbers can be manipulated consistently, and the extent to which they have the expected properties. Although one goal of this report is to show the development of real analysis without recourse to actual infinities, there is no problem with the

computable manipulation of finite symbols *representing* infinities.

6.4.2 Predicates:

$$\text{nonstandard}_Q \Rightarrow \text{zero?}_Z \circ \text{den}_Q$$

$$\text{standard?}_Q \Rightarrow \text{not} \circ \text{nonstandard}_Q$$

$$\text{infinite?}_Q \Rightarrow \text{and} \circ (\text{nonstandard}_Q, \text{not} \circ \text{zero?}_Z \circ \text{num}_Q)$$

$$\text{indefinite?}_Q \Rightarrow \text{and} \circ (\text{nonstandard}_Q, \text{zero?}_Z \circ \text{num}_Q)$$

Comments: Thus, infinite? $\infty \Leftrightarrow$ **true**, $\text{nonstandard}_Q\ \infty \Leftrightarrow$ **true**, etc.

6.4.3 Standard Equality:

$$\text{eq}_Q\ (m/n)\ \infty \Leftrightarrow \text{eq}_Q\ (m/n)\ -\infty \Leftrightarrow \text{zero?}_N\ n$$

$$\text{eq}_Q\ \infty\ -\infty \Leftrightarrow \text{eq}_Q\ -\infty\ \infty \Leftrightarrow \textbf{true}$$

Comments: These results follow from the definition of 'eq_Q', and reflect the fact that that definition is only appropriate for finite rational numbers. In the following subsection we extend the equality operation to accommodate nonstandard rationals.

6.4.4 Nonstandard Equality:

$$\text{eq}_{Q+} \Rightarrow \lambda(qr)\begin{cases}\text{and (infinite? } q\text{) (infinite? } r\text{)} \rightarrow \text{equiv (pos?}_Z \circ \text{num}_Q\ q\text{) (pos?}_Z \circ \text{num}_Q\ r\text{)} \\ \textbf{else}\ \text{eq}_Q\ q\ r\end{cases}$$

TABLE 1. Equality Relation on Nonstandard Rationals

eq_{Q+}	r	∞	$-\infty$	χ
q	$\text{eq}_Q\ q\ r$	**false**	**false**	**true**
∞	**false**	**true**	**false**	**true**
$-\infty$	**false**	**false**	**true**	**true**
χ	**true**	**true**	**true**	**true**

Comments: This operation has the properties shown in Table 1. Notice that the indefinite quantity χ is equal to all quantities, finite or infinite. Therefore this relation is not transitive, since 0/1 equals χ, and χ equals 1/1, but 0/1 is not equal to 1/1. On the other hand, the infinities (∞, $-\infty$) are unequal to all finite quantities and are unequal to each other. Henceforth, we use '=' as an abbreviation for 'eq_{Q+}'. That is, '$q = r$' means '$\text{eq}_{Q+}\ q\ r$'.

6.4.5 Nonnegative:

$$\text{nonneg?}_Q\ \infty \Leftrightarrow \textbf{true}$$

$$\text{nonneg?}_Q\ -\infty \Leftrightarrow \textbf{false}$$

$$\text{nonneg?}_Q\ \chi \Leftrightarrow \textbf{true}$$

Comments: These properties follow directly from the definitions:

$$\begin{aligned}\text{nonneg?}_Q\ \infty &\Rightarrow \text{nonneg?}_Q\ (+1/\ +0)\\ &\Rightarrow \text{equiv}\ (\text{neg?}_Z\ +1)\ (\text{neg?}_Z\ +0)\\ &\Rightarrow \text{equiv}\ \textbf{false}\ \textbf{false}\\ &\Leftrightarrow \textbf{true}\end{aligned}$$

Similarly,

$$\text{nonneg?}_Q\ -\infty \Rightarrow \text{equiv}\ (\text{neg?}_Z\ -1)\ (\text{neg}_Z\ +0) \Leftrightarrow \textbf{false}$$

$$\text{nonneg?}_Q\ \chi \Rightarrow \text{equiv}\ (\text{neg?}_Z\ +0)\ (\text{neg?}_Z\ -0) \Leftrightarrow \textbf{true}$$

Comments: As expected, ∞ is nonnegative and $-\infty$ is negative. That χ, the indefinite quantity, is nonnegative may seem incorrect. This situation is a result of our considering all zeros to be nonnegative, including -0. Otherwise we could define a "negative indefinite" quantity $-\chi \Rightarrow -0/\ +0$. It can be seen that it is reasonable to consider χ nonnegative, since χ is equal to every rational number, including zero. Hence, it is nonnegative, in the sense that it is greater than *or equal to* zero.

6.4.6 Negative:

$$\text{neg?}_Q\ \infty \Rightarrow \textbf{false}$$

$$\text{neg?}_Q\ -\infty \Rightarrow \textbf{true}$$

$$\text{neg?}_Q\ \chi \Rightarrow \textbf{false}$$

6.4.7 Nonstandard Addition:

$$\text{sum}_{Q+} \Rightarrow \lambda(qr)\begin{cases}\text{and}\ (\text{eq}_{Q+}\ q\ r)\ [\text{and}\ (\text{infinite?}\ q)\ (\text{infinite?}\ r)] \rightarrow q\\ \textbf{else}\ \text{sum}_Q\ q\ r\end{cases}$$

Comments: The properties of this operation are shown in Table 2. These follow from the definitions, for example

TABLE 2. Addition of Nonstandard Rationals

sum_Q	r	∞	$-\infty$	χ
q	$q + r$	∞	$-\infty$	χ
∞	∞	∞	χ	χ
$-\infty$	$-\infty$	χ	$-\infty$	χ
χ	χ	χ	χ	χ

$$m/n + 1/0 = (m \cdot 0 + n \cdot 1)/(n \cdot 0) = m/0 = \infty \text{ if } m > 0$$

$$m/n + -1/0 = (m \cdot 0 + n \cdot -1)/(n \cdot 0) = -m/0 = -\infty \text{ if } m > 0$$

$$m/n + 0/0 = (m \cdot 0 + n \cdot 0)/(n \cdot 0) = 0/0 = \chi$$

A nonstandard addition is necessary if the operation is to have the expected properties, otherwise we would have $\infty + \infty = \chi$, since

$$\infty + \infty = 1/0 + 1/0 = (1 \cdot 0 + 0 \cdot 1)/(0 \cdot 0) = 0/0 = \chi$$

6.4.8 Negation:

$$\text{neg}_Q\ \infty \Leftrightarrow -\infty$$

$$\text{neg}_Q\ -\infty \Leftrightarrow \infty$$

$$\text{neg}_Q\ \chi \Leftrightarrow \chi$$

6.4.9 Difference:

The properties of subtraction with respect to the nonstandard quantities follows directly from those of addition by the identity

$$\text{dif}_Q\ q\ r = \text{sum}_Q\ q\ (\text{neg}_Q\ r)$$

6.4.10 Ordering: $\text{lt}_{Q+} \Rightarrow \text{neg?}_Q : \text{sum}_{Q+} : (1\text{st}, \text{neg}_Q \circ 2\text{nd})$

TABLE 3. Ordering of Nonstandard Rationals

lt_{Q+}	r	∞	$-\infty$	χ
q	$q < r$	**true**	**false**	**false**
∞	**false**	**false**	**false**	**false**
$-\infty$	**true**	**true**	**false**	**false**
false	**false**	**false**	**false**	**false**

Comments: The properties of this operation are shown in Table 3. As expected we have $-\infty < q < \infty$ by this ordering. Notice that χ is neither greater than q nor less than q, for it is equal to q, for every q, finite or infinite.

6.4.11 Products, Reciprocals and Quotients:

TABLE 4. Multiplication of Nonstandard Rationals

prod_Q	0	r	∞	$-\infty$	χ
0	0	0	χ	χ	χ
q	0	qr	$\pm\infty$	$\mp\infty$	χ
∞	χ	$\pm\infty$	∞	$-\infty$	χ
$-\infty$	χ	$\mp\infty$	$-\infty$	∞	χ
χ	χ	χ	χ	χ	χ

The properties of multiplication of nonstandard rationals, as deduced from the standard definition of rational multiplication, are shown in Table 4. The sign $\pm$ must be taken in the same sense as the sign of q (or r), whereas the sign $\mp$ must be taken in the opposite sense.

The nonstandard properties of the reciprocal are:

$\text{recip}_Q\ \infty\ =\ \text{recip}_Q\ -\infty\ =\ 0$

$\text{recip}_Q\ \chi\ =\ \chi$

etc.

The properties of the quotient follow from the identity

$$\text{quo}_Q\ q\ r\ =\ \text{prod}_Q\ q\ (\text{recip}_Q\ r)$$

7. Real Numbers

In this chapter we reach the culmination of our development, defining real numbers as convergent computable rational sequences of a particular kind. Operations on real numbers are defined as computable operations on these sequences.

7.1 Primitive Ideas

7.1.1 Definition of a Regular Sequence:

A sequence of rational numbers x is called *regular* if and only if for any nonzero natural numbers m and n we have

$$\mathrm{le}_Q\ [\mathrm{abs}_Q\ (\mathrm{dif}_Q\ x_m\ x_n)]\ [\mathrm{sum}_Q\ (+1/\ m)\ (+1/\ n)]\ \Leftrightarrow\ \mathbf{true}$$

Intuitively, the sequence x is regular if and only if

$$|\ x_m\ -\ x_n\ |\ \leqslant m^{-1} + n^{-1}$$

for all natural m, $n > 0$. Note that for any *given* m and n it is finitely decidable whether this condition holds. However, it is not in general finitely decidable for arbitrary m and n.

7.1.2 Definition of Real Number:

A *real number* is a regular sequence of rational numbers.

7.1.3 Equality:

Two real numbers x and y are *equal*, written $x = y$, if and only if for any nonzero natural number n we have

$$\mathrm{le}_Q\ [\mathrm{abs}_Q\ (\mathrm{dif}_Q\ x_n\ y_n)]\ [\mathrm{prod}_Q\ (+2/\ n)]\ \Leftrightarrow\ \mathbf{true}$$

Comments: Intuitively, $x = y$ if and only if

$$|\ x_n\ -\ y_n\ |\ \leqslant 2n^{-1}$$

for all natural $n > 0$.

Note that while this condition is finitely decidable for a given n, it is not generally finitely decidable for arbitrary n.

7.1.4 Nonstandard Objects: There are two alternative notions of real numbers. First, real numbers can be considered the equivalence classes under the equality relation of regular sequences of rational numbers. Second, real numbers can be identified with the regular sequences themselves. In this case we find that there are many different (nonstandard) real numbers that are treated as equal by the arithmetic operations and the equality relation.

We can also consider the nonstandard real number all of whose rational approximations are ∞. To see if this object fits the definition of a real number we must see if the difference between its m and nth rational approximations is greater than or equal to $1/m + 1/n$. Now, the difference between its mth and nth approximations is:

$$|\infty - \infty| = |\chi| = \chi$$

We have previously seen that χ is greater than or equal to every rational number, since it is equal to every rational number. Thus,

$$|\infty - \infty| = \chi \leqslant 1/m + 1/n$$

and so the sequence composed of all ∞s is a real number. The same applies for $-\infty$. The following operations work, or can be extended to work, on infinite reals in much the same way as the rational operations.

7.1.5 Creation of Reals from Rationals: $\text{real}_R \Rightarrow \lambda x(\lambda nx)$

Comments: This function converts a rational number into a real all of whose rational approximations are the same:

$$\text{real}_R\ r \Rightarrow \lambda x(\lambda nx)\, r \Rightarrow \lambda nr$$

Now notice that the kth rational approximation of $\text{real}_R\ r$ is:

$$(\text{real}_R\ r)_k \Rightarrow \text{real}_R\ r\ k \Rightarrow \lambda nr\ k \Rightarrow r$$

7.1.6 Real Denotations:

$0.0 \Rightarrow \text{real}_R\ (0/\ 1)$

$1.0 \Rightarrow \text{real}_R\ (1/\ 1)$

7.1.7 Example Normal Form: For concreteness we exhibit the normal form of the real number 0.0:

$$
\begin{aligned}
0.0 &\Rightarrow \mathrm{real}_R\ (0/\ 0) \\
&\Rightarrow \lambda n\,(0/\ 0) \\
&\Rightarrow \lambda n\,(\lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)x\ \bot\ \bot) \\
&\qquad \lambda s\,(s\ \lambda(xy)x\ \bot\ \bot))) \\
&\quad \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)y\ \lambda s\,(s\ \lambda(xy)x\ \bot\ \bot) \\
&\qquad \lambda s\,(s\ \lambda(xy)x\ \bot\ \bot)))\ \lambda s\,(s\ \lambda(xy)x\ \bot\ \bot)))
\end{aligned}
$$

7.1.8 Example Normal Form of Irrational Number: We have said that this computational approach to real analysis represents all objects as finite structures. To illustrate this we will show the finite structure *that represents* an irrational number, the square root of 2. The square root of two will be a regular rational sequence s such that s_k is the kth rational approximation of the square root. By Newton's Method observe that

$$s_k \;=\; \tfrac{1}{2}(2\ /\ s_{k-1}\ +\ s_{k-1}),\ \ k\ >\ 1$$

The initial approximation, s_1 is 1. Thus the sequence can be defined recursively by the rule:

$$\mathbf{rec}\ \ s\ \ =\ \ \lambda k\begin{cases} k=1 \rightarrow 1 \\ \mathbf{else}\ \tfrac{1}{2}(2/\ s_{k-1}+s_{k-1}) \end{cases}$$

Since this sequence converges quadratically it is regular and, hence, represents a real number. To show that s is indeed a finite formula in the lambda calculus, we rewrite it using only the notational abbreviations introduced so far:

$$\mathbf{rec}\ \ s\ \ =\ \ \lambda k\begin{cases} (\mathrm{eq}_N\ 1\ k)\ \rightarrow\ 1/\ 1 \\ \mathbf{else}\ [\mathrm{prod}_Q\ (1/\ 2)\ (\mathrm{sum}_Q\ \{\mathrm{quo}_Q\ (2/\ 1)\ [s\ (\mathrm{pred}_N\ k)]\}\ [s\ (\mathrm{pred}_N\ k)])] \end{cases}$$

The above formula is finite; it will remain finite when all the abbreviations are eliminated, since each symbol is rewritten by a finite string. We illustrate the first few steps of this process (we have replaced the name 's' by '$\sqrt{2}$'):

$$\sqrt{2} \Rightarrow \mathbf{Y}\ \lambda s\ \lambda k \begin{cases} (\mathrm{eq}_N\ 1\ k) \rightarrow 1/\ 1 \\ \textbf{else}\ [\mathrm{prod}_Q\ (1/\ 2)\ (\mathrm{sum}_Q\ \{\mathrm{quo}_Q\ (2/\ 1)\ [s\ (\mathrm{pred}_N\ k)]\}\ [s\ (\mathrm{pred}_N\ k)])] \end{cases}$$

$$\Rightarrow \mathbf{Y}\ \lambda s\ \lambda k\ \{(\mathrm{eq}_N\ 1\ k)\ (1/\ 1)\ [\mathrm{prod}_Q\ (1/\ 2)$$

$$(\mathrm{sum}_Q\ \{\mathrm{quo}_Q\ (2/\ 1)\ [s\ (\mathrm{pred}_N\ k)]\}\ [s\ (\mathrm{pred}_N\ k)])]\}$$

$$\Rightarrow \mathbf{Y}\ \lambda s\ \lambda k\ \{(\mathrm{not:ne}_N\ 1\ k)\ (\mathrm{rat}_Q\ 1\ 1)\ [\mathrm{prod}_Q\ (\mathrm{rat}_Q\ 1\ 2)$$

$$(\mathrm{sum}_Q\ \{\mathrm{quo}_Q\ (\mathrm{rat}_Q\ 2\ 1)\ [s\ (\mathrm{pred}_N\ k)]\}\ [s\ (\mathrm{pred}_N\ k)])]\}$$

By eliminating all the abbreviations we would arrive at a seminormal form expression representing the square root of 2. The full seminormal form of the square root of 2 is shown in the Appendix.

7.1.9 Rational Approximation: $\mathrm{approx}_R \Rightarrow \mathrm{conv\ Id}$

Comments: The function '$\mathrm{approx}_R\ n$' gives then n th rational approximation of a real number. That is, if x is a real number then '$\mathrm{approx}_R\ n\ x$' is its n th rational approximation:

$$\mathrm{approx}_R\ n\ x \Rightarrow \mathrm{conv\ Id}\ n\ x \Rightarrow \mathrm{Id}\ x\ n \Rightarrow x\ n \Leftrightarrow x_n$$

7.2 Arithmetic

7.2.1 Sum: $\mathrm{sum}_R \Rightarrow \lambda(xy)[\mathrm{sum}_Q \circ (x,y) \circ (\mathrm{prod}_N\ 2)]$

Comments: This definition makes the n th rational approximation of $x+y$ the sum of the $2n$ th rational approximations of x and y:

$$(x+y)_n = x_{2n} + y_{2n}$$

Now note that:

$$(\mathrm{sum}_R\ x\ y)_n \Rightarrow \mathrm{sum}_R\ x\ y\ n$$

$$\Rightarrow [\mathrm{sum}_Q \circ (x,y) \circ (\mathrm{prod}_N\ 2)]\ n$$

$$\Rightarrow \mathrm{sum}_Q \circ (x,y)\ [(\mathrm{prod}_N\ 2)\ n]$$

$$\Rightarrow \mathrm{sum}_Q \circ (x,y)\ (\mathrm{prod}_N\ 2\ n)$$

$$\Rightarrow \mathrm{sum}_Q\ [x\ (\mathrm{prod}_N\ 2\ n)]\ [y\ (\mathrm{prod}_N\ 2\ n)]$$

It is easy to show that $x+y$ is a real number and that this operation has the expected properties (commutativity, associativity, etc.).

7.2.2 Maximum: $\max_R \Rightarrow \Phi\, \max_Q$

Comments: The n th rational approximation of the maximum of two reals is the maximum of their n th rational approximations:

$$(\max_R\ x\ y)_n \Rightarrow \max_R\ x\ y\ n \Rightarrow \Phi\, \max_Q\ x\ y\ n \Rightarrow \max_Q\ x_n\ y_n$$

7.2.3 Negation: $\mathrm{neg}_R \Rightarrow \mathbf{B}\, \mathrm{neg}_Q$

Comments: The n th rational approximation of the negation of a real is the negation of its n th rational approximation:

$$(\mathrm{neg}_R\ x)_n \Rightarrow \mathrm{neg}_R\ x\ n \Rightarrow \mathbf{B}\, \mathrm{neg}_Q\ x\ n \Rightarrow \mathrm{neg}_Q\ (x\ n) \Leftrightarrow \mathrm{neg}_Q\ x_n$$

7.2.4 Minimum: $\min_R \Rightarrow \mathrm{neg}_R : (\Psi\, \max_R\ \mathrm{neg}_R)$

Comments: This is based on the fact that the minimum of x and y is the negation of the maximum of the negations of x and y:

$$\min_R\ x\ y \Rightarrow \mathrm{neg}_R : (\Psi\, \max_R\ \mathrm{neg}_R)\ x\ y$$
$$\Rightarrow \mathrm{neg}_R\ (\Psi\, \max_R\ \mathrm{neg}_R\ x\ y) \Rightarrow \mathrm{neg}_R\ [\max_R\ (\mathrm{neg}_R\ x)\ (\mathrm{neg}_R\ y)]$$

7.2.5 Absolute Value: $\mathrm{abs}_R \Rightarrow \mathbf{S}\, \max_R\ \mathrm{neg}_R$

Comments: The absolute value of a number is the maximum of the number and its negation:

$$\mathrm{abs}_R\ x \Rightarrow \mathbf{S}\, \max_R\ \mathrm{neg}_R\ x \Rightarrow \max_R\ x\ (\mathrm{neg}_R\ x)$$

7.2.6 Canonical Bound: $\mathrm{cbd}_R \Rightarrow \mathrm{ceil}_Q \circ [\mathrm{sum}_Q\ (2/\ 1)] \circ \mathrm{abs}_Q\ (\mathrm{approx}_R\ 1)$

Comments: The canonical bound of a real is used in the definition of the product, which follows. The canonical bound of x is the least integer that is two greater than the absolute value of the first rational approximation of x:

$$\mathrm{cbd}_R\ x \Rightarrow \{\mathrm{ceil}_Q \circ [\mathrm{sum}_Q\ (2/\ 1)] \circ \mathrm{abs}_Q\ (\mathrm{approx}_R\ 1)\}\ x$$
$$\Rightarrow \mathrm{ceil}_Q\ \{\mathrm{sum}_Q\ (2/\ 1)\ [\mathrm{abs}_Q\ (\mathrm{approx}_R\ 1\ x)]\}$$
$$\Rightarrow \mathrm{ceil}_Q\ [\mathrm{sum}_Q\ (2/\ 1)\ (\mathrm{abs}_Q\ x_1)]$$

The latter represents the ceiling of $2 + |x_1|$.

7.2.7 Product:

$$\text{prod}_R \;\Rightarrow\; \lambda(xy)\left\{\text{prod}_Q \circ (x,y) \circ (\text{prod}_N\ 2) \circ \text{mag}_Z \circ [\text{prod}_Z\ (\Psi\ \max_Z\ \text{cbd}_R\ x\ y)]\right\}$$

Comments: The n th rational approximation of xy is the product of the $2kn$ th rational approximations of x and y, in which k is the maximum of the canonical bounds of x and y. First notice that

$$\Psi\ \max_Z\ \text{cbd}_R\ x\ y \;\Rightarrow\; \max_Z\ (\text{cbd}_R\ x)\ (\text{cbd}_R\ y) \;\Leftrightarrow\; k$$

Therefore,

$$(\text{prod}_R\ x\ y)_n \;\Rightarrow\; \text{prod}_R\ x\ y\ n$$

$$\Rightarrow\; [\text{prod}_Q \circ (x,y) \circ (\text{prod}_N\ 2) \circ \text{mag}_Z \circ (\text{prod}_Z\ k)]\ n$$

$$\Rightarrow\; \text{prod}_Q \circ (x,y)\ \{\text{prod}_N\ 2\ [\text{mag}_Z\ (\text{prod}_Z\ k\ n)]\}$$

$$\Rightarrow\; \text{prod}_Q \circ (x,y)\ (2kn)$$

$$\Rightarrow\; \text{prod}_Q\ (x\ 2kn)\ (y\ 2kn)$$

$$\Leftrightarrow\; \text{prod}_Q\ x_{2kn}\ y_{2kn}$$

7.2.8 Exponentiation to Natural Power: $\text{expt}_R \;\Rightarrow\; \lambda(nx)[\text{rpt}\ (\text{prod}_R\ x)\ n\ 1]$

Comments: The basis for this is:

$$\text{expt}_R\ n\ x \;\Rightarrow\; \text{rpt}\ (\text{prod}_R\ x)\ n\ 1 \;\Rightarrow\; (\text{prod}_R\ x)^n\ 1 \;=\; x^n$$

We usually write 'x^n' for '$\text{expt}_R\ n\ x$'.

7.2.9 Notation:

Since each of the classes of objects we have defined effectively includes the preceding, and the operations and relations of the latter classes extend the former classes, we will drop the subscripts (N, Z, Q and R) and functional notation, and henceforth use conventional mathematical notation. Thus we will write '$x + 2y$' for

$$\text{sum}_R\ x\ \{\text{prod}_R\ [\text{real}_R\ (2/\ 1)]\ y\}$$

7.3 Predicates and Relations

7.3.1 Positive with Modulus: pos? $\Rightarrow \lambda(nx)[\mathrm{gt}_Q\ x_n\ (1/\ n)]$

Comments: If n is a nonzero natural number and x is a real number, then x is called *positive with modulus* n if and only if $x_n > 1/\ n$. The argument n names a place where x_n deviates sufficiently from zero. Notice that this predicate is algorithmically decidable; that is, if x and n are defined then pos? n x is defined.

7.3.2 Positive: We say that a real number is *positive* if and only if there is a nonzero natural n such that the real is positive with modulus n. That is, x is positive if and only if there is a nonzero natural n such that pos? n x $\Leftrightarrow$ **true**.

Comments: Note that it is not in general algorithmically decidable whether or not a real number is positive. Thus, it is not safe to treat a real number as a positive unless we have calculated (i.e., constructed algorithmically) the modulus n required by the above definition. This algorithmic undecidability is inherited by all the other relations.

7.3.3 Negative with Modulus: neg? $\Rightarrow$ pos?:(1st, neg $\circ$ 2nd)

Comments: Here x is negative with modulus n if and only if $-x$ is positive with modulus n:

$$\text{neg? } n\ x \Rightarrow \text{pos?:(1st, neg} \circ \text{2nd)}\ n\ x \Rightarrow \text{pos? } n\ (\text{neg } x)$$

Notice that if $-x$ is positive (with modulus n), then $(-x)_n > 1/\ n$. Therefore, x is negative (with modulus n) if and only if $x_n < -1/\ n$.

7.3.4 Zero with Modulus Predicate: zero? $\Rightarrow$ not $\circ$ or $\circ$ (pos?,neg?)

Comments: A number is zero (with modulus n) if and only if it is neither positive nor negative (with modulus n):

$$\begin{aligned}\text{zero? } x &\Rightarrow \text{not} \circ \text{or} \circ (\text{pos?,neg?})\ x \\ &\Rightarrow \text{not}[\text{or} \circ (\text{pos?,neg?})\ x] \\ &\Rightarrow \text{not}[\text{or (pos? } x)\ (\text{neg? } x)]\end{aligned}$$

Notice that x is zero with modulus n if and only if $-1/\ n \leq x_n \leq 1/\ n$, that is, $|\ x_n\ | \leq 1/\ n$.

7.3.5 Equality with Modulus: eq $\Rightarrow$ λn [(zero? n):dif]

Comments: Two reals are equal with modulus n if and only if their difference is zero with modulus n :

$$\text{eq } n \; x \; y \;\Rightarrow\; (\text{zero? } n):\text{dif } x \; y \;\Rightarrow\; \text{zero?}(\text{dif } x \; y)$$

Notice that since

$$| \, x - y \, |_n \;=\; | \, (x - y)_n \, | \;=\; | \, x_{2n} - y_{2n} \, |$$

we know that x equals y with modulus n if and only if $| \, x_{2n} - y_{2n} \, | \leq 1/n$. We will generally write '$x =_n y$' to mean that x equals y with modulus n. Finally, notice that x equals y if and only if for *all* nonzero naturals n, $x =_n y$.

7.3.6 Nonnegative: A real number x is called *nonnegative* if and only if for all nonzero natural numbers n we have $x_n \geqslant -1/n$.

Comments: This is also in general algorithmically undecidable, since it depends on an infinite process, *viz.*, checking that the above condition applies for all n.

7.3.7 Greater-Than with Modulus: gt $\Rightarrow$ λn [(pos? n):dif]

Comments: We say that x is greater then y with modulus n when $x - y$ is positive with modulus n :

$$\text{gt } n \; x \; y \;\Rightarrow\; (\text{pos? } n):\text{dif } x \; y \;\Rightarrow\; \text{pos? } n \; (\text{dif } x \; y)$$

Hence, since

$$(x - y)_n \;=\; x_{2n} - y_{2n}$$

we have x is greater than y with modulus n if and only if $x_{2n} - y_{2n} > 1/n$, that is, $x_{2n} > y_{2n} + 1/n$. Notice that this relation is algorithmically decidable, but that it requires supplying the modulus n, which names the place where x sufficiently exceeds y. We will generally write $x >_n y$ to mean that x is greater than y with modulus n.

7.3.8 Greater-Than Relation: We say x is greater than y and write $x > y$ if and only if $x - y$ is positive.

Comments: That is, if and only if there is a nonzero natural number n such that x is greater than y

with modulus n:

$$\text{gt } n \; x \; y \iff \textbf{true}$$

This relation is not algorithmically decidable.

7.3.9 Less-Than with Modulus: lt $\Rightarrow$ λn [conv (gt n)]

Comments: Observe

$$\text{lt } n \; x \; y \Rightarrow \text{conv (gt } n \text{)} \; x \; y \Rightarrow \text{gt } n \; y \; x .$$

Thus, $x_{2n} + 1/n < y_{2n}$.

7.3.10 Less-Than Relation: We say x is less than y and write $x < y$ if and only if $y > x$.

7.3.11 Greater-or-Equal with Modulus: ge $\Rightarrow$ λn [or:(gt n, eq n)]

Comments: To see this, observe:

$$\text{ge } n \; x \; y \Rightarrow \text{or:(gt } n \text{, eq } n \text{)} \; x \; y$$
$$\Rightarrow \text{or (gt } n \; x \; y \text{) (eq } n \; x \; y \text{)}$$

Comments: Note that x is greater than or equal to y with modulus n if and only if $x_{2n} + 1/n \geq y_{2n}$.

7.3.12 Greater-or-Equal Relation: We say x is greater than or equal to y and write $x \geq y$ if and only if $x - y$ is nonnegative.

7.3.13 Less-or-Equal with Modulus: le $\Rightarrow$ λn [conv (ge n)]

Comments: Note that x is less than or equal to y with modulus n if and only if $x_{2n} \leq y_{2n} + 1/n$.

7.3.14 Less-or-Equal Relation: We say x is less than or equal to y and write $x \leq y$ if and only if $x \geq y$.

7.3.15 Inequality with Modulus: ne $\Rightarrow$ λn [not:(eq n)]

Comments: Observe that

$$\text{ne } n \; x \; y \Rightarrow \text{not:(eq } n \text{)} \; x \; y \Rightarrow \text{not(eq } n \; x \; y \text{)}$$

Comments: Note that x is not equal to y with modulus n if and only if $|\, x_{2n} - y_{2n} \,| > 1/\, n$.

7.3.16 Inequality Relation: We say x is unequal to y and write $x \neq y$ if and only if either $x > y$ or $x < y$.

7.4 Reciprocals and Quotients

7.4.1 Reciprocal: If x and y are real numbers, then x is called a *reciprocal of* y if and only if $xy = 1$.

Comments: The following three sections define a function for computing the reciprocal of a real number.

7.4.2 Zero Bound: zerobnd $\Rightarrow \lambda(nx)|\text{floor}_Q\ (\text{recip}_Q\ \{\text{quo}_Q\ [\text{dif}_Q\ (\text{abs}_Q\ x_n)\ (1/\, n)]\ (2/\, 1)\})|$

Comments: If x is nonzero with modulus n, then zerobnd n x has the following property. Let N = zerobnd n x. Then, for all $m \geqslant N$, $x_m \geqslant 1/\, N$.

7.4.3 Reciprocal with Modulus:

$$\text{recip} \Rightarrow \lambda(nx)\ \lambda k \begin{cases} k < N \rightarrow 1/\, (x_{N\,3}) \\ k \geqslant N \rightarrow 1/\, (x_{kN\,2}) \end{cases}$$

where N = zerobnd n x

Comments: Notice that the reciprocal is computable only if a modulus of nonnullity is known. Intuitively,

$$(\text{recip } n\ x)_k = 1/\, (x_{N\,3}), \text{ if } k < N$$

$$= 1/\, (x_{kN\,2}), \text{ if } k \geqslant N$$

We will generally write 'x^{-1}' for 'recip n x', omitting the modulus of nonnullity. It is important to observe that x^{-1} is computable only if we can compute this modulus.

7.4.4 Quotient with Modulus: quo $\Rightarrow \lambda n\ \{\text{prod:}[1st, (\text{recip } n) \circ 2\text{nd}]\}$

Comments: Therefore, the quotient is the product of the first and the reciprocal of the second:

$$\text{quo } x\ y \ \Rightarrow\ \text{prod:}[\text{1st}, (\text{recip } n) \circ \text{2nd}]\ x\ y \ \Rightarrow\ \text{prod } x\ (\text{recip } n\ y) \ =\ xy^{-1}$$

We will generally write '$x\ /\ y$' or '$\frac{x}{y}$' for 'quo $n\ x\ y$'. Notice that $x\ /\ y$ is computable only if a modulus of nonnullity for y is known. The modulus is implicit in the division notation.

7.5 Important Properties of the Real Numbers

7.5.1 Approximation Theorem: The nth rational approximation of a real x approximates x to within $1/\ n$. That is, for all nonzero naturals n,

$$|\ x\ -\ x_n\ |\ \leqslant 1/\ n$$

7.5.2 Rational Between Two Reals: $\text{meanrat} \ \Rightarrow\ \lambda(nxy)[\text{prod}_Q\ \frac{1}{2}\ (\text{sum}_Q\ x_{2n}\ y_{2n})]$

Comments: Notice that if x and y are reals and n is a nonzero natural number, then the result of (meanrat $n\ x\ y$) is a *rational* number with the value

$$\text{meanrat } n\ x\ y \ \Leftrightarrow\ \tfrac{1}{2}(x_{2n}\ +\ y_{2n})$$

Since (meanrat $n\ x\ y$) is the average of the $2n$th rational approximations of x and y, it is between these approximations. We will generally write '$\text{meanrat}_n(x,y)$' for 'meanrat $n\ x\ y$'.

7.5.3 Denseness of Rational Numbers: If x and y are real numbers with $x < y$, then there exists a rational number r such that $x < r < y$.

Proof: If $x < y$ then there is some nonzero natural n such that $x < y$ with modulus n. We now claim that

$$r \ =\ \text{meanrat}_n(x,y)$$

is a rational number satisfying $x < r < y$. That r is a rational follows from the definition of meanrat. To show $x < r$ we must show $r\ -\ x$ is positive. Therefore, recalling that $(x\ -\ y)_n = x_{2n}\ -\ y_{2n}$, we have

$$\begin{aligned} r\ -\ x\ &\geqslant\ r\ -\ x_{2n} \\ &\geqslant\ r\ -\ x_{2n}\ -\ |\ x_{2n}\ -\ x\ | \\ &=\ \tfrac{1}{2}(x_{2n}\ +\ y_{2n})\ -\ x_{2n}\ -\ |\ x_{2n}\ -\ x\ | \end{aligned}$$

$$= \frac{1}{2}(y_{2n} - x_{2n}) - |x_{2n} - x|$$

By the approximation theorem:

$$r - x \geqslant \frac{1}{2}(y_{2n} - x_{2n}) - (2n)^{-1}$$

Since $x < y$ with modulus n, we know $y_{2n} - x_{2n} > n^{-1}$, therefore

$$r - x > \frac{1}{2} n^{-1} - (2n)^{-1} = 0$$

Therefore $x < r$. A similar derivation shows that $r < y$.

7.5.4 Cantor's Theorem: Let R be any sequence of rational numbers and u and v any real numbers with $u < v$. Then there exists a real number ξ between x and y, $u \leqslant \xi \leqslant v$, such that for all nonzero naturals n we have $\xi \neq R_n$.

Proof: Since $u < v$ there is some nonzero natural number κ such that is the modulus of this relation, $u <_\kappa v$. We will construct sequences of rationals x and y such that the following properties hold:

1. For all natural $m \geqslant n \geqslant 1$,

$$u \leqslant x_n \leqslant x_m < y_m \leqslant y_n \leqslant v$$

   That is, the x_ns are less than the y_ns, and they are always growing closer together.

2. For all nonzero natural n, either $x_n >_\kappa R_n$ or $y_n <_\kappa R_n$ (or both). That is, the x_ns and y_ns are all sufficiently different from the corresponding R_ns.

3. For all nonzero natural n, $y_n - x_n < 1/n$. That is, x and y are equal (in the sense of real numbers). We will define $\xi = x = y$.

We construct the sequences x and y using a "diagonalization" function 'diag', which constructs the two sequences in parallel. If $S = \text{diag}(\kappa, u, v)$, then $x = S_0$ and $y = S_1$. Hence we introduce 'x' and 'y' as explicit abbreviations:

'x' $\Rightarrow$ 'diag κ u v 0'

'y' $\Rightarrow$ 'diag κ u v 1'

The definition of the diagonalization function is:

$$\textbf{rec}\ \ \mathrm{diag}\,(\kappa, u, v, b) \;=\; \lambda n$$

$$\begin{cases} b = 0 \rightarrow \begin{cases} n = 0 \;\rightarrow\; u \\ R_n >_\kappa x_{n-1} \;\rightarrow\; \mathrm{meanrat}_\kappa\lfloor x_{n-1},\ \min(R_n,\ y_{n-1})\rfloor \\ R_n <_\kappa y_{n-1} \;\rightarrow\; \mathrm{meanrat}_\kappa\lfloor \max(R_n,\ x_{n-1},\ y_n - 1/\ n),\ y_n\rfloor \end{cases} \\ b = 1 \rightarrow \begin{cases} n = 0 \;\rightarrow\; v \\ R_n >_\kappa x_{n-1} \;\rightarrow\; \mathrm{meanrat}_\kappa\lfloor x_n,\ \min(R_n,\ y_{n-1},\ x_n + 1/\ n)\rfloor \\ R_n <_\kappa y_{n-1} \;\rightarrow\; \mathrm{meanrat}_\kappa\lfloor \max(R_n,\ x_{n-1}),\ y_{n-1}\rfloor \end{cases} \end{cases}$$

It is necessary to show that the sequences x and y defined by this function have the above required properties (1) - (3).

Consider x_n and suppose $R_n >_\kappa x_{n-1}$. Then,

$$x_n \;=\; \mathrm{meanrat}_\kappa\lfloor x_{n-1},\ \min(R_n,\ y_{n-1})\rfloor$$

Therefore,

$$x_{n-1} \;<\; x_n \;<\; \min(R_n,\ y_{n-1}) \;\leqslant\; y_{n-1}$$

Similarly,

$$y_n \;=\; \mathrm{meanrat}_\kappa\lfloor x_n,\ \min(R_n,\ y_{n-1},\ x_n + 1/\ n)\rfloor$$

Hence,

$$x_n \;<\; y_n \;<\; \min(R_n,\ y_{n-1},\ x_n + 1/\ n) \;\leqslant\; y_{n-1}$$

(Notice that y_n is constrained to be within $1/\ n$ above x_n.)

On the other hand, suppose that $R_n <_\kappa y_{n-1}$. Then,

$$y_n \;=\; \mathrm{meanrat}_\kappa\lfloor \max(R_n,\ x_{n-1}),\ y_{n-1}\rfloor$$

Therefore,

$$x_{n-1} \;\leqslant\; \max(R_n,\ x_{n-1}) \;<\; y_n \;<\; y_{n-1}$$

Similarly,

$$x_n \;=\; \mathrm{meanrat}_\kappa\lfloor \max(R_n,\ x_{n-1},\ y_n - 1/\ n),\ y_n\rfloor$$

Hence,

$$x_{n-1} \leqslant \max(R_n, x_{n-1}, y - 1/n) < x_n < y_n$$

(Notice that in this case x_n is constrained to be within $1/n$ below y_n.) Combining these results we have

$$u = x_0 < x_1 < \cdots < x_n < y_n < \cdots < y_1 < y_0 = v$$

Thus condition (1) is satisfied. Also notice that since x_n and y_n are always constrained to be within $1/n$ of each other, $y_n - x_n < 1/n$, and condition (3) is satisfied.

To see that (2) holds, consider any step in the construction of x and y. Since $x_{n-1} < y_{n-1}$, either $R_n >_\kappa x_{n-1}$ or $R_n <_\kappa y_{n-1}$ (or both). If $R_n >_\kappa x_{n-1}$ then

$$x_n < \min(R_n, y_{n-1}) \leqslant R_n$$

so $x_n < R_n$. Similarly $y_n < R_n$. On the other hand, if $R_n <_\kappa y_{n-1}$ then

$$R_n \leqslant \max(R_n, x_{n-1}) < y_n$$

so $R_n < y_n$. Similarly $R_n < x_n$. Thus we have that either $x_n > R_n$ or $y_n < R_n$.

The rest of the proof is simple. First we must show that x and y are real numbers. Hence, suppose that $m \geqslant n$. By properties (1) and (3) we have

$$|x_m - x_n| = x_m - x_n < y_n - x_n < 1/n < 1/m + 1/n$$

Thus x is a real number; the result is analogous for y. Furthermore, by (3) we know that $x = y$. Therefore, let $\xi = x = y$.

To complete the proof we must show that the diagonal number ξ does not appear in the sequence R. That is, for all nonzero natural n, $\xi \neq R_n$. Let n be chosen. By (2) either $x_n > R_n$ or $y_n < R_n$. Suppose $x_n > R_n$. Then, since (1) implies $x > x_n$ we know $x > R_n$ and hence $\xi = x \neq R_n$. On the other hand suppose that $y_n < R_n$. Since (1) implies $y < y_n$ we know $y < R_n$ and hence $\xi = y \neq R_n$. So, in either case $\xi \neq R_n$, and the theorem is proved.

Comments: Cantor's Theorem is usually interpreted as saying that the real numbers cannot be enumerated. Indeed, we have shown here how given any enumeration R of (computable) real numbers we can compute a real ξ not in that enumeration. But this is paradoxical, since we clearly can enumerate all lambda calculus formulas: just enumerate all finite strings of symbols in the (finite) alphabet, and

strike out those that do not fit the syntactic rules of the lambda calculus. Since this enumeration will include all those formulas that compute real numbers, it seems that we have the enumeration of the (computable) reals that Cantor's Theorem denies. This is in effect the famous Löwenheim-Skolem Paradox.

We can escape from the paradox by looking carefully at the premises of Cantor's Theorem, which require that we be given a sequence R of real numbers. Recall that a sequence of real numbers is defined as a lambda calculus formula R such that for every nonzero natural number n, R_n is defined (i.e., has a seminormal form), and R_n is a real number. Now, our enumeration of lambda calculus formulas is not a sequence of reals, since it generates many formulas that do not fit the definition of a real number, and may not even have a seminormal form. Furthermore, we cannot convert our enumeration of lambda calculus formulas into a sequence of reals by striking out those formulas that are not reals, since this property is algorithmically undecidable.

This leads us to the following interpretation of Cantor's Theorem: It is impossible to define a computable enumeration that generates *all* the computable reals and *only* the computable reals.

8. Calculus

8.1 Convergence, Limits and Series

8.1.1 Convergence:

If X is a sequence of reals and N is a sequence of natural numbers, then we say that X *converges to* y *with modulus* N if and only if for all nonzero natural numbers k and n such that $n \geqslant N_k$ we have

$$| X_n - y | \leqslant k^{-1}$$

When N or y is not relevant we may omit mentioning them and say, for example, X *converges* or X *is convergent.*

8.1.2 Limit:

$\lim_N X \Rightarrow \mathbf{S}\,[X \circ (\text{limmod } N)]\ (\text{prod } 2)$

where limmod $\Rightarrow \lambda N\,\{\max \circ [\text{prod } 3,\ N \circ (\text{prod } 3)]$

Comments: Intuitively, we have:

$$\lim_N X = \lambda k\,[(X_{M_k})_{2k}] \quad \textbf{where} \quad M = \lambda k\,[\max(3k,\ N_{3k})]$$

That is, the k th rational approximation of $\lim_N X$ is the $2k$ th rational approximation of X_{M_k}, where the k th rational approximation of M is the maximum of $3k$ and N_{3k}. To see that our definition gives this, let M = limmod N and observe:

$(\lim_N X)_k \Rightarrow \lim_N X\ k$

$\Rightarrow \mathbf{S}\,[X \circ (\text{limmod } N)]\ (\text{prod } 2)\ k$

$\Rightarrow [X \circ (\text{limmod } N)]\ k\ (\text{prod } 2\ k)$

$\Rightarrow X\ (\text{limmod } N\ k)\ (\text{prod } 2\ k)$

$= X\ (M\ k)\ (2k)$

$= (X_{M_k})_{2k}$

Now we check the definition of 'limmod':

$M_k \Rightarrow$ limmod $N\ k$

$\Rightarrow$ max $\circ$ [prod 3, $N \circ$ (prod 3)] k

$\Rightarrow$ max (prod 3 k) [N (prod 3 k)]

$=$ max $3k\ N_{3k}$

Notice that the limit of a convergent sequence is always computable, provided that the modulus of convergence is supplied.

8.1.3 Convergence to Limits: If the sequence of reals X converges with modulus N, then X converges to $\lim_N X$ (with modulus N).

Proof: There are two parts to the proof. First we must show that $\lim_N X$ is a real number; second we must show that X converges to $\lim_N X$.

Let $y = \lim_N X$. To show that y is a real number we have to show that for any nonzero rationals m, n we have

$$| y_m - y_n | \leqslant m^{-1} + n^{-1}$$

Without loss of generality assume that $m \geqslant n$ and recall that $M_k = \max(3k, N_{6k})$. By the triangle inequality, the definition of y and the approximation theorem we have:

$$\begin{aligned} | y_m - y_n | &\leqslant | y_m - X_{M_m} | + | X_{M_m} - X_{M_n} | + | X_{M_n} - y_n | \\ &= | (X_{M_m})_{2m} - X_{M_m} | + | X_{M_m} - X_{M_n} | + | X_{M_n} - (X_{M_n})_{2n} | \\ &\leqslant (2m)^{-1} + | X_{M_m} - X_{M_n} | + (2n)^{-1} \end{aligned}$$

Since X converges with modulus N, it converges to some real number with this modulus. Call this number z. Since X converges to z with modulus N we know that for any $n \geqslant N_k$ we have $| X_n - z | \leqslant k^{-1}$. Since $M_m \geqslant N_{3m}$ (by the construction of M), we know

$$| X_{M_m} - z | \leqslant (3m)^{-1}$$

Similarly for $M_n \geqslant N_{3n}$:

$$| X_{M_n} - z | \leqslant (3n)^{-1}$$

Combining these two inequalities with the previous inequality yields

$$\begin{aligned} | y_m - y_n | &\leqslant (2m)^{-1} + (3m)^{-1} + (3n)^{-1} + (2n)^{-1} \\ &\leqslant m^{-1} + n^{-1} \end{aligned}$$

Therefore y is a real number.

Now we must show that X converges to y; in fact we will show that it converges with modulus M. Therefore, suppose that $n \geqslant M_k$; we must show that $| y - X_n | \leqslant k^{-1}$. By the triangle inequality, the approximation theorem, the definition of y and the convergence of X we have:

$$\begin{aligned} | y - X_n | &\leqslant | y - y_n | + | y_n - X_{M_n} | + | X_{M_n} - X_n | \\ &\leqslant n^{-1} + | (X_{M_n})_{2k} - X_{M_n} | + | X_{M_n} - X_n | \\ &\leqslant n^{-1} + (2n)^{-1} + | X_{M_n} - X_n | \\ &\leqslant n^{-1} + (2n)^{-1} + | X_{M_n} - z | + | X_n - z | \end{aligned}$$

Now, since $M_n \geqslant N_{3n}$ and $n \geqslant M_k \geqslant N_{3k}$ we have

$$| y - X_n | \leqslant n^{-1} + (2n)^{-1} + (3n)^{-1} + (3k)^{-1}$$

Now, since $n \geqslant M_k \geqslant 3k$ we know

$$\begin{aligned} | y - X_n | &\leqslant (3k)^{-1} + (6k)^{-1} + (9k)^{-1} + (3k)^{-1} \\ &= (1/3 + 1/6 + 1/9 + 1/3)k^{-1} \\ &= (17/18)k^{-1} \\ &< k^{-1} \end{aligned}$$

Thus X converges to $y = \lim_N X$ with modulus M. But since we were given that X converges to its limit with modulus N, we know that X converges to $\lim_N X$ with modulus N.

8.1.4 Properties of Limits: Let X and Y be sequences, then

$$\begin{aligned} \lim \text{sum} \circ (X, Y) &= \text{sum}(\lim X, \lim Y) \\ \lim \text{dif} \circ (X, Y) &= \text{dif}(\lim X, \lim Y) \\ \lim \text{prod} \circ (X, Y) &= \text{prod}(\lim X, \lim Y) \\ \lim \text{quo} \circ (X, Y) &= \text{quo}(\lim X, \lim Y) \end{aligned}$$

That is, the limit of the sum is the sum of the limits, etc.

8.1.5 Compact Interval: A pair of reals $[a,b]$ is called a *compact interval* if and only if $a \leqslant b$.

Comments: When we say that a real x is in the compact interval $[a,b]$ we mean that $a \leqslant x \leqslant b$. Notice that it is not in general algorithmically decidable whether a pair of reals is a compact interval, or whether a real is in a given compact interval. We will occasionally use other interval notations with their obvious meanings. For example, we say that x is in the half-open interval $(a,b]$ if and only if $a < x \leqslant b$.

8.1.6 Compact Interval with Modulus: A pair of real numbers $[a,b]$ is called a *compact interval with modulus* n if and only if a is less than or equal to b with modulus n. We say that a real x is in the compact interval provided that a is less than or equal to x with modulus n, and x is less than or equal to b with modulus n.

Comments: If the modulus n is provided, then it becomes algorithmically decidable whether a pair of reals is a compact interval, and whether a given real belongs to the interval.

8.1.7 Notation for Limits: When the modulus of convergence N is understood or implied, we will write 'lim X' instead of 'lim$_N$ X'. We also permit the following abbreviations:

$$\lim_{n \to \infty} f(n) \Rightarrow \lim f$$

$$\lim_{n \to -\infty} g(n) \Rightarrow \lim_{n \to \infty} g(-n)$$

The first of these limits exists provided that there are N and y such that for all $k > 0$ and all $m \geqslant N(k)$ we have $| f(m) - y | \leqslant 1/k$. The second exists provided that there are N and y such that for all $k < 0$ and all $m \leqslant N(k)$ we have $| g(m) - y | \leqslant -1/k$.

If f is defined on $(0,1]$ and g is defined on $[-1,0)$ then we can write:

$$\lim_{x \to 0+} f(x) \Rightarrow \lim_{n \to \infty} f(1/n)$$

$$\lim_{x \to 0-} g(x) \Rightarrow \lim_{x \to 0+} g(-x)$$

The first of these exists provided that there are δ and y such that for all $\epsilon > 0$ and all $0 \leqslant \varsigma \leqslant \delta(\epsilon)$ we have $| f(\varsigma) - y | \leqslant \epsilon$. The existence condition for the second is analogous.

If $\lim_{x \to 0+} f(x) = \lim_{x \to 0-} f(x)$, then we can write $\lim_{x \to 0} f(x)$ to denote their common limit.

If f is defined on $(y, y+1]$ and g is defined on $[y-1, y)$ then we can write:

$$\lim_{x \to a+} f(x) \Rightarrow \lim_{\delta \to 0+} f(a + \delta)$$

$$\lim_{x \to a-} g(x) \Rightarrow \lim_{\delta \to 0-} f(a + \delta)$$

The first of these exists provided that there are δ and y such that for all $\epsilon > 0$ and all $0 \leq \varsigma \leq \delta(\epsilon)$ we have $| f(a + \varsigma) - y | \leq \epsilon$. The existence condition for the second is analogous. We use $\lim_{x \to a} f(x)$ to denote the common limit when the limits approaching from above and below a are equal.

If f is defined on the compact interval $I = [u, v]$ then we can write:

$$_I\lim_{x \to a+} f(x) \Rightarrow \lim_{\delta \to 0+} f[a + \delta(v - a)]$$

$$_I\lim_{x \to a-} f(x) \Rightarrow \lim_{\delta \to 0-} f[a + \delta(a - u)]$$

We will omit mention of the interval I and the direction from which the limit is approached when these are not relevant. Again, we use $_I\lim_{x \to a} f(x)$ for the common limit.

8.1.8 Partial Sums of Series: $\mathbf{rec}\ \Sigma\ X = \lambda n \begin{cases} n = 0 \to 0 \\ \mathbf{else}\ \Sigma\ X\ (n-1) \end{cases}$

Comments: A *series* is a sequence that is meant to be summed. If X is a series then ΣX is the sequence of partial sums of X. That is:

$$(\Sigma X)_0 = 0$$

$$(\Sigma X)_1 = X_1$$

$$(\Sigma X)_2 = X_2 + X_1$$

$$\vdots$$

$$(\Sigma X)_n = X_n + X_{n-1} + \cdots + X_2 + X_1$$

8.1.9 Infinite Sums of Series: $\sum^{\infty} \Rightarrow \lim \cdot \Sigma$

Comments: If X is a convergent series, then $\sum^{\infty} X$ is the sum of all its terms:

$$\sum^{\infty} X \;\Rightarrow\; \lim \circ \Sigma\, X \;\Rightarrow\; \lim(\Sigma\, X) \;=\; \lim_{n \to \infty} (\Sigma X)_n$$

The modulus of convergence here is implicit; if we must be explicit about it we will write:

$$\sum^{\infty}_{N} \;\Rightarrow\; \lim_{N} \circ\, \Sigma$$

8.1.10 Exponential Function: $\exp \;\Rightarrow\; \lambda x\, [\sum^{\infty} \text{quo} \circ (\text{conv expt } x, \text{fac})]$

Comments: The basis for this definition is $\exp x \;=\; \sum^{\infty} \lambda n\, (x^n / n!)$

8.2 Continuous Functions

8.2.1 Continuity on a Compact Interval: If f and ω are real-valued functions and f is defined on a compact interval I, then we call f ***continuous on I with modulus of continuity ω*** if and only if for each positive ϵ we have

$$|\, f(x) - f(y) \,| \;\leqslant \epsilon$$

whenever x and y are in I and $|\, x - y \,| \;\leqslant \omega(\epsilon)$.

Comments: We often omit mention of the modulus of continuity when it is not relevant. Notice that it is not in general algorithmically decidable whether a function is continuous, since that decision would require testing the above condition for every x and y in the interval.

8.2.2 Continuity on an Arbitrary Interval: We say that a real-valued function defined on an arbitrary interval is ***continuous*** on that interval provided that it is continuous on every compact subinterval of the given interval.

8.3 Differentiation

8.3.1 Differentiability: If f, g and δ are real-valued functions, and f and g are continuous on a compact interval $[a, b]$ with $a < b$, then we say ***g is a derivative of f on $[a, b]$ with modulus of differentiability δ*** if and only if for all positive ϵ we have

$$|\, f(y) - f(x) - g(x)(y - x) \,| \;\leqslant\; \epsilon\, |\, y - x \,|$$

whenever x and y are in $[a, b]$ and $|\, y - x \,| < \delta(\epsilon)$. A function f is said to be ***differentiable*** on an

interval if and only if there is a function g that is its derivative on that interval.

8.3.2 Derivative: $D \Rightarrow \lambda(If)\,\lambda x\left[{}_I\lim_{y \to x} \frac{f(y) - f(x)}{y - x}\right]$

Comments: Let $g = D_I f$; then

$$g(x) = {}_I\lim_{y \to x} \frac{f(y) - f(x)}{y - x}$$

Notice that the derivative of a function is always computable, as is the result of evaluating that derivative on any point in the interval.

8.3.3 Differentiation Theorem: If f is differentiable on an interval I then $D_I f$ is a derivative of f on that interval. If f has two derivatives, then they are equal.

Proof: Since f is differentiable on I we know that there g continuous on I and there exists δ such that for all $\epsilon > 0$ and all x, y in I for which $| y - x | < \delta(\epsilon)$ we have

$$| f(y) - f(x) - g(x)(y - x) | \leqslant \epsilon | y - x |$$

Now, for any x in I let

$$h(x) = (D_I f)(x) = {}_I\lim_{y \to x} F(y),$$

where $F(y) = [f(y) - f(x)] / (y - x)$.

We must show that the limit $h(x)$ exists and is equal to $g(x)$.

Notice that the definition of F includes a division, which is computable only if we know a modulus of nonnullity. We can eliminate this problematic division be rewriting the limit:

$${}_I\lim_{y \to x} F(y) \Rightarrow \lim_{\delta \to 0+} F[x + \delta(b - x)] \Rightarrow \lim_{n \to \infty} F[x + (b - x)/n]$$

Now notice that

$$F[x + (b - x)/n] = \frac{f[x + (b - x)/n] - f(x)}{(b - x)/n} = n\,\frac{f[x + (b - x)/n] - f(x)}{b - x}$$

The troublesome division by $y - x$ has been converted to a safe multiplication by n. (The division by $b - x$ is not troublesome because its modulus of nonnullity does not have to vary during the limiting

process.)

We now show that $F(y)$ converges to $g(x)$ as $y \to x$. Let an $\epsilon > 0$ be chosen; we must find a $y > x$ such that $|F(y) - g(x)| \leq \epsilon$. Take $y < x + \delta(\epsilon)$, where δ is the modulus of differentiability. Since f is differentiable we know

$$|f(y) - f(x) - g(x)(y - x)| \leq \epsilon |y - x|$$

Dividing both sides by $y - x > 0$ yields

$$\left| \frac{f(y) - f(x)}{y - x} - g(x) \right| \leq \epsilon$$

That is, $|F(y) - g(x)| \leq \epsilon$. *Q.E.D.*

8.3.4 Properties of the Derivative: The derivative of the sum is the sum of the derivatives:

$$D[\text{sum} \circ (f, g)] = \text{sum} \circ (Df, Dg)$$

The derivative of the product is the sum of the product of the first times the derivative of the second, and the product of the second times the derivative of the first:

$$D[\text{prod} \circ (f, g)] = \text{sum} \circ [\text{prod} \circ (f, Dg), \text{prod} \circ (g, Df)]$$

The derivative of the reciprocal of a function is the quotient of the negative derivative of the function divided by the square of the function.

$$D(\text{recip} \circ f) = \text{quo} \circ [\text{neg} \circ Df, (\text{expt } 2) \circ f]$$

The derivative of the identity function is the constant 1 function:

$$D\ \text{Id} = \mathbf{K}\ 1$$

The derivative of a constant function is the constant 0 function:

$$D\ (\mathbf{K}\ c) = \mathbf{K}\ 0$$

8.3.5 Repeated Differentiation: $D^n \Rightarrow \text{rpt } n\ D$

Comments: For example,

$$D^2 f \Rightarrow \text{rpt } 2\ D\ f \Rightarrow D(D\ f)$$

The conventional notations 'dF / dx' and '$D_x F$' are equivalent to '$D\ \lambda x F$'.

8.3.6 Partial Differentiation: $\partial_k^n \Rightarrow \lambda f\ \{\lambda(x_1 \cdots x_n)[D\ \lambda x(f\ x_1 \cdots x_{k-1}\ x\ x_k \cdots x_n)]\ x_k\}$

Comments: Thus ∂_k^n is the partial derivative of an n-adic function with respect to its kth argument. For example, if $g = \partial_2^3 f$ then

$$g(u,v,w) = D[\lambda x(f\ u\ x\ w)](v)$$

Thus the conventional notation '$\partial F / \partial y$', where F is a function of x, y and z in that order, is equivalent to '$\partial_2^3\ \lambda(xyz)F$'.

8.4 Integration

8.4.1 Quadrature: quad $\Rightarrow \lambda(f\,a\,b\,n)\left[\frac{b-a}{n}\sum_{i=0}^{n-1} f\left(a + i\frac{b-a}{n}\right)\right]$

Comments: Thus quad $f\ a\ b\ n$ is the quadrature of f from a to b with n steps.

8.4.2 Integral: $\int \Rightarrow \lambda(f\,a\,b)[\lim(\text{quad}\ f\ a\ b)]$

Comments: Therefore,

$$\int f\ a\ b \Rightarrow \lim(\text{quad}\ f\ a\ b) \Rightarrow \lim_{n\to\infty}\left[\frac{b-a}{n}\sum_{i=0}^{n-1} f\left(a + i\frac{b-a}{n}\right)\right]$$

We usually write '$\int_a^b f$' for '$\int f\ a\ b$'. Thus the conventional notation '$\int_a^b F\ dx$' is equivalent to '$\int_a^b \lambda x F$'.

8.4.3 Reversed Range of Integration: $\int_a^b f \Rightarrow \int_b^a f$, if $a > b$.

8.4.4 Properties of Definite Integral:

The integral of the sum is the sum of the integrals:

$$\int_a^b \text{sum} \circ (f,g) = \text{sum}\left(\int_a^b f, \int_a^b g\right)$$

The integral of a constant times the function is the constant times the integral of the function:

$$\int_a^b \text{prod} \circ (\mathbf{K}\alpha, f) = \text{prod}\left(\alpha, \int_a^b f\right)$$

The integral from a to b is the integral from a to c plus the integral from c to b:

$$\int_a^b f \;=\; \int_a^c f \;+\; \int_c^b f$$

8.4.5 Indefinite Integral:

Notice that $\int f$ is an indefinite integral of f in the following sense: If $g = \int f$, then $g(x,y) = \int_x^y f$. Another notion of indefinite integral is $\int_a f$, since if $h = \int_a f$ then $h(x) = \int_a^x f$.

8.4.6 Properties of Indefinite Integral:

$$\int_a \text{sum} \circ (f, g) \;=\; \text{sum} \circ (\int_a f, \int_a g)$$

$$\int_a \text{prod} \circ (\mathbf{K}\alpha, f) \;=\; \text{prod} \circ (\mathbf{K}\alpha, \int_a f)$$

$$\int_a f \;=\; \text{sum} \circ [\mathbf{K}(\int_a^b f), \int_b f]$$

8.4.7 Fundamental Theorem of Calculus:

If f is continuous on a proper interval I and a is a point of I, then

$$D(\int_a f) \;=\; f$$

That is, the derivative of the indefinite integral (of the second kind) is the original function. Furthermore, if h is any differentiable function such that $Dh = f$, then for some real α,

$$\text{dif} \circ (\int_a f, h) \;=\; \mathbf{K}\,\alpha$$

That is, the difference of $\int_a f$ and h is a constant function.

8.4.8 Natural Logarithm: $\ln \;\Rightarrow\; \int_1 \text{recip}$

Comments: Therefore,

$$\ln x \;\Rightarrow\; \int_1 \text{recip}\; x \;\Rightarrow\; \int_1^x \text{recip} \;=\; \int_1^x x^{-1} dx$$

8.4.9 Definite Integral of N-adic Function:

$$_k^n\!\int_a^b f \;\Rightarrow\; \lambda(x_1 \cdots x_{n-1})[\int_a^b \lambda x\,(f\; x_1 \cdots x_{k-1}\; x\; x_k \cdots x_{n-1})]$$

Comments: Thus $_k^n\!\int_a^b f$ is the definite integral of the n-adic function f, taking its kth argument from a to b. Notice that the definite integral of an n-adic function is and $(n-1)$-adic function. For example, if $g = {}_2^3\!\int_a^b f$, then

$$g(u,v) = \int_a^b \lambda x [f(u,x,v)]$$

Thus, the conventional notation '$\int_a^b F\ dy$', where F is a function of x, y and z in that order, is equivalent to '${}_2^3\int_a^b \lambda(zyz)F$'.

8.4.10 Indefinite Integral of N-adic Function:

$${}_k^n\int_a f \Rightarrow \lambda(x_1 \cdots x_n)[\int_a \lambda x (f\ x_1 \cdots x_{k-1}\ x\ x_{k+1} \cdots x_n)\ x_k]$$

Comments: As expected, the indefinite integral of an n-adic function is an n-adic function. For example, if $g = {}_2^3\int_a f$, then

$$g(u,v,w) = \int_a^e \lambda x [f(u,x,w)]$$

Appendix 1: Seminormal Form of Square-Root of 2

We exhibit the first few pages of a seminormal form for $\sqrt{2}$ computed by the formula derived in the body of the report. Ellipses ($\cdots$) indicate where major portions of the formula have been omitted; however, they look about the same as the portions shown below. The complete formula is 1974379 symbols in length, which would occupy about 4700 pages in the format used below. It must be emphasized again that the formula, although large, is *finite*.

The formula was computed by converting the lambda calculus definitions into the corresponding LISP program shown in Appendix 2. The length of the formula was computed in two ways: (1) by direct counting (using a LISP program) of the output of Appendix 2, (2) by converting the Appendix 2 program into the related length computation program shown in Appendix 3.

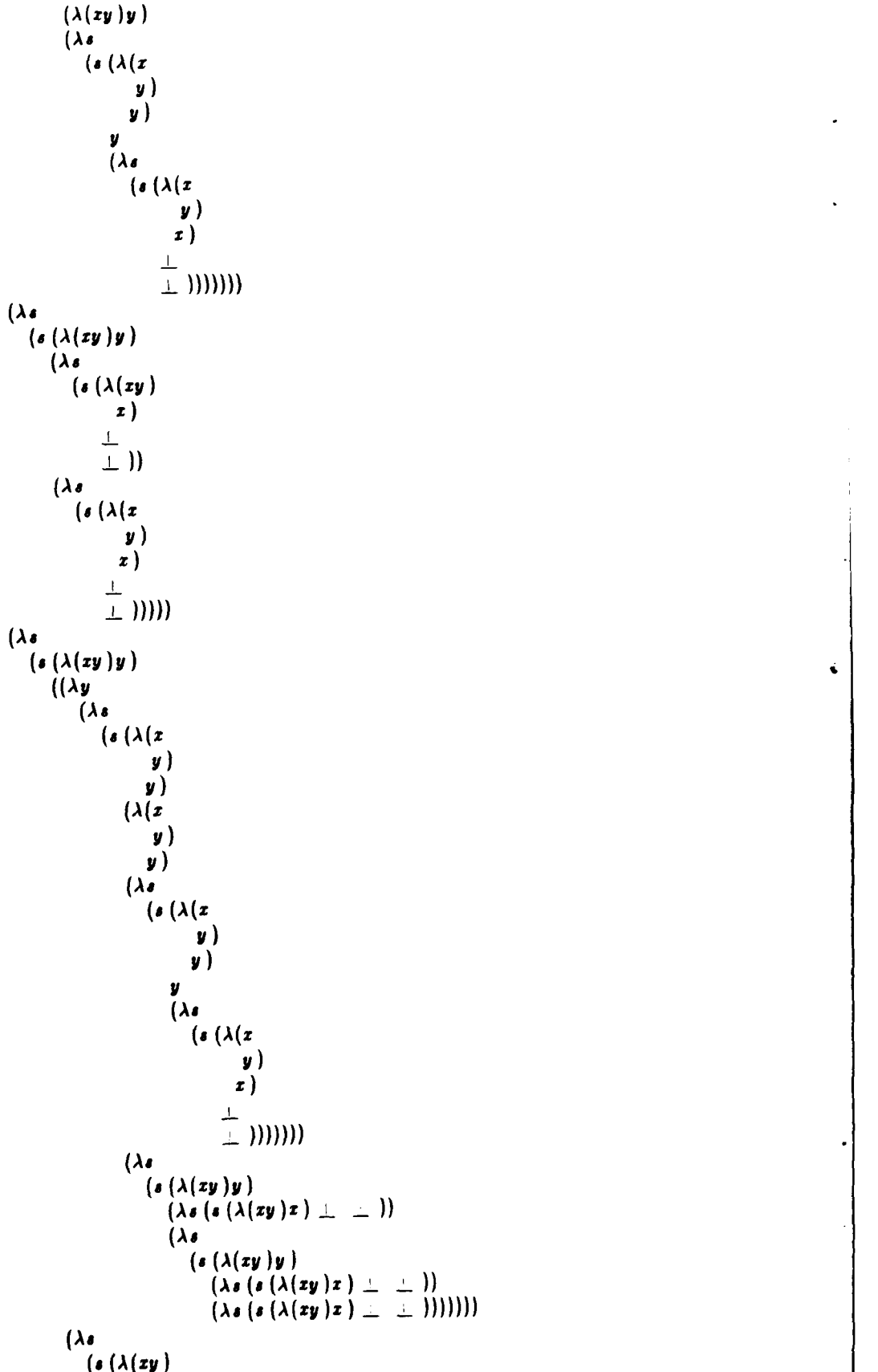
Several recurring features are visible in the portion of the formula shown here. The expressions '$(\lambda(xy)x)$' and '$(\lambda(xy)y)$' represent the truth values **true** and **false**. Formulas beginning '$(\lambda s\ (s\ \cdots))$' are list values, some of which end with '$\perp\ \perp$' and hence represent null lists. The formulas '$(\lambda(xyz)x)$', '$(\lambda(xyz)y)$' and '$(\lambda(xyz)z)$' represent the three basic selector operations on lists: null, hd and tl.

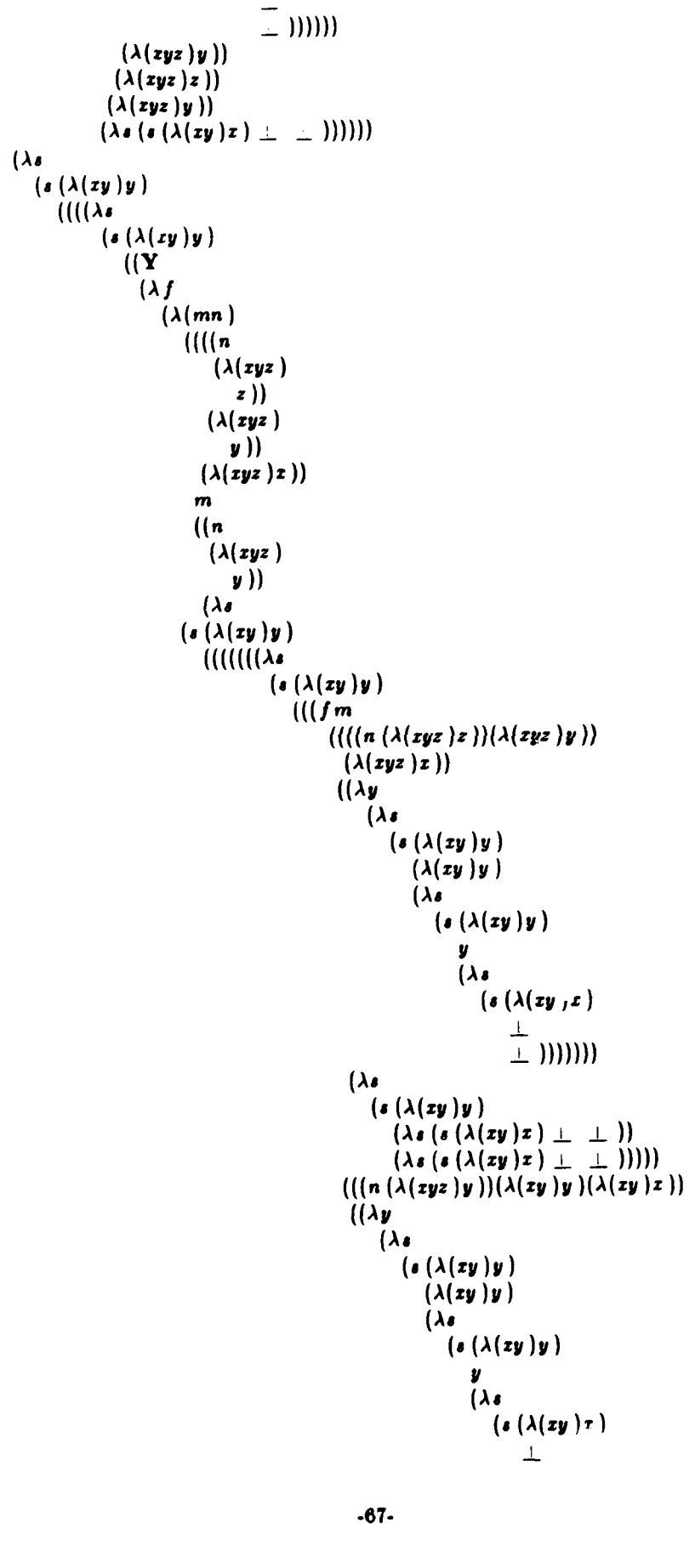
$\sqrt{2} \Rightarrow$

```
(Y
 (λs
  (λk
   (((((Y
        (λf
         (λ(mn)
          ((n (λ(xyz)x))
           (λ(xy)y)
           ((m (λ(xyz)x))
            (λ(xy)x)
            (f (m (λ(xyz)z))(n (λ(xyz)z))))))))
       (λs
        (s (λ(xy)y)
         (λs (s (λ(xy)x) ⊥ ⊥))
         (λs (s (λ(xy)x) ⊥ ⊥))))
       k)
      (λ(xy)x)
      ((Y
        (λf
         (λ(mn)
          ((n (λ(xyz)x))
           (λ(xy)y)
           ((m (λ(xyz)x))
            (λ(xy)x)
            (f (m (λ(xyz)z))(n (λ(xyz)z))))))))
       k
       (λs
        (s (λ(xy)y)
         (λs (s (λ(xy)x) ⊥ ⊥))
         (λs (s (λ(xy)x) ⊥ ⊥))))
       (λ(xy)y)
       (λ(xy)x))
      (λs
       (s (λ(xy)y)
        ((λy
          (λs
           (s (λ(xy)y)
```

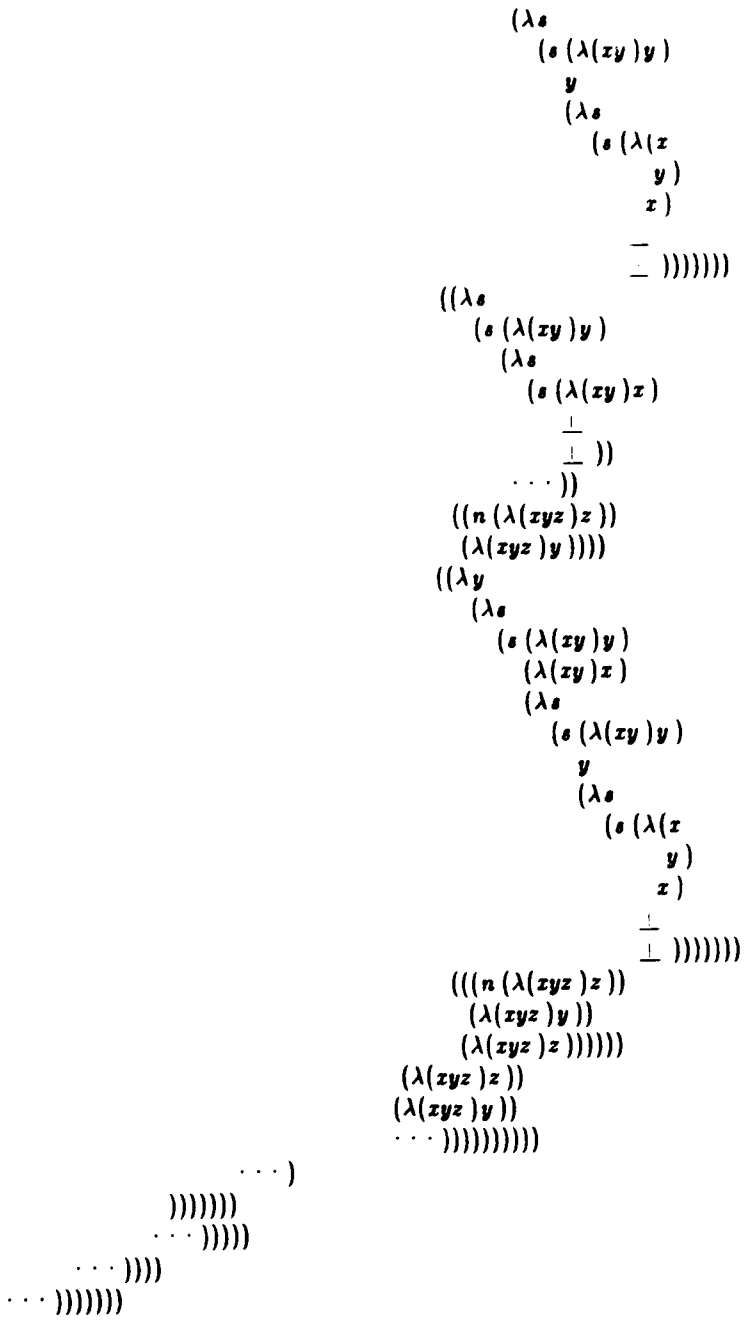
```
                (λ(xy)y)
                (λs
                  (s(λ(xy)y)
                    y
                    (λs
                      (s(λ(xy)x)⊥ ⊥ )))))))
          (λs
            (s(λ(xy)y)
              (λs(s(λ(xy)x)⊥ ⊥ ))
              (λs(s(λ(xy)x)⊥ ⊥ )))))
          (λs
            (s(λ(xy)y)
              ((λy
                 (λs
                   (s(λ(xy)y)
                     (λ(xy)y)
                     (λs
                       (s(λ(xy)y)
                         y
                         (λs
                           (s(λ(xy)x)
                             ⊥
                             ⊥ )))))))
               (λs
                 (s(λ(xy)y)
                   (λs(s(λ(xy)x)⊥ ⊥ ))
                   (λs(s(λ(xy)x)⊥ ⊥ )))))
               (λs(s(λ(xy)x)⊥ ⊥ ))))))
(λs
  (s(λ(xy)y)
    (((λs
        (s(λ(xy)y)
          ((((λs
              (s(λ(xy)y)
                ((λy
                   (λs
                     (s(λ(xy)y)
                       (λ(xy)y)
                       (λs
                         (s(λ(xy)y)
                           y
                           (λs
                             (s(λ(x
                                  y)
                                x)
                               ⊥
                               ⊥ )))))))
                 (λs
                   (s(λ(xy)y)
                     (λs
                       (s(λ(xy)x)
                         ⊥
                         ⊥ ))
                     (λs
                       (s(λ(xy)x)
                         ⊥
```

```
                    _ )))))
          (λs
            (s (λ(xy )y )
              ((λy
                (λs
                  (s (λ(xy )y )
                    (λ(xy )y )
                    (λs
                      (s (λ(x
                            y )
                          y )
                        y
                        (λs
                          (s (λ(x
                                y )
                              x )
                            ⊥
                            ⊥ )))))))
                (λs
                  (s (λ(xy )y )
                    (λs
                      (s (λ(xy )
                          x )
                        ⊥
                        ⊥ ))
                    (λs
                      (s (λ(x
                            y )
                          y )
                        (λs
                          (s (λ(x
                                y )
                              x )
                            ⊥
                            ⊥ ))
                        (λs
                          (s (λ(x
                                y )
                              x )
                            ⊥
                            ⊥ )))))))
                (λs
                  (s (λ(xy )x )
                    ⊥
                    ⊥ ))))))
          (λ(xyz )y ))
          (λ(xyz )y ))
          (λ(xy )y )
          (λ(xy )x ))
          (λs
            (s (λ(xy )y )
              ((((λs
                  (s (λ(xy )y )
                    ((λy
                      (λs
                        (s (λ(xy )y )
```





```
                                    ⊥ )))))))
                  ((λs
                     (s (λ(xy)y)
                       (λs (s (λ(xy)x) ⊥ ⊥ ))
                        · · · ))
                   ((n (λ(xyz)z))(λ(xyz)y))))
                  ((λy
                     (λs
                       (s (λ(xy)y)
                         (λ(xy)x)
                         (λs
                           (s (λ(xy)y)
                             y
                             (λs
                               (s (λ(xy)x)
                                 ⊥
                                 ⊥ )))))))
                   (((n (λ(xyz)z))(λ(xyz)y))
                    (λ(xyz)z))))))
                (λ(xyz)y))
               (λ(xy)y)
               (λ(xy)x))
              (λs
                (s (λ(xy)y)
                  (((f m
                     ((((n (λ(xyz)z))(λ(xyz)y))
                       (λ(xyz)x))
                      ((λy
                         (λs
                           (s (λ(xy)y)
                             (λ(xy)y)
                             (λs
                               (s (λ(xy)y)
                                 y
                                 (λs
                                   (s (λ(xy)
                                       x)
                                     ⊥
                                     ⊥ )))))))
                       (λs
                         (s (λ(xy)y)
                           (λs
                             (s (λ(xy)x)
                               ⊥
                               ⊥ ))
                           (λs
                             (s (λ(xy)x)
                               ⊥
                               ⊥ )))))
                       (((n (λ(xyz)y))
                         (λ(xy)y)
                         (λ(xy)x))
                        ((λy
                           (λs
                             (s (λ(xy)y)
                               (λ(xy)y)
```



Appendix 2: LISP Program to Compute Seminormal Form of Square-Root 2

```
(setq false   '(& (x y) y))

(setq true    '(& (x y) x))

(defun Not (x)    (list x false true))

(defun Or (x y)    (list x true y))

(defun Equiv (x y)    (list x y (Not y)))

(setq 1of3    '(& (x y z) x))

(setq 2of3    '(& (x y z) y))

(setq 3of3    '(& (x y z) z))

(defun List (n h r)    (list '& 's (list 's n h r)))

(setq Nil    (List true 'undef 'undef))

(defun Cons (h r)    (List false h r))

(defun Null (x)    (list x 1of3))

(defun hd (x)    (list x 2of3))

(defun tl (x)    (list x 3of3))

(defun pair (x y)    (Cons x (Cons y Nil)))

(setq n0    Nil)

(setq n1  (Cons Nil Nil))

(setq n2    (Cons Nil (Cons Nil Nil)))

(defun Nzero? (n)   (Null n))

(defun Nsucc (n)    (Cons Nil n))

(setq cNsucc (list '& 'n (Cons Nil 'n)))

(defun Npred (x)   (tl x))

(setq rpt    (list 'Y (list '& 'r (list '& '(f n a)
                              (list (Nzero? 'n) 'a
                                    (list 'f (list 'r 'f (Npred 'n) 'a)))))))

(defun Nsum  (m)  (list rpt cNsucc m))

(defun Nprod (m n)    (list rpt (Nsum m) n n0))
```

```
(setq cNlt    (list 'Y (list '& 'f (list '& '(m n) (list (Nzero? 'n) false
                                  (list (Nzero? 'm) true
                                    (list 'f (Npred 'm) (Npred 'n))))))))

(defun Nlt (m n)    (list cNlt m n))

(defun Ngt (m n)    (Nlt n m))

(defun Nne (x y)    (Or (Nlt x y) (Ngt x y)))

(defun Neq (x y)    (Not (Nne x y)))

(defun Zplus (n)    (pair false n))

(defun Zminus (n)    (pair true n))

(setq p1    (Zplus n1))

(setq p2    (Zplus n2))

(defun Zmag (n)    (hd (tl n)))

(defun Zminus? (n)    (hd n))

(defun Zplus? (n)    (Not (Zminus? n)))

(defun Zzero? (n)    (Nzero? (Zmag n)))

(defun Zneg (m)    (pair (Zplus? m) (Zmag m)))

(defun Zsucc (n)    (list (Zzero? n) p1
                          (list (Zplus? n) (Zplus (Nsucc (Zmag n)))
                                (Zminus (Npred (Zmag n))))))

(defun Zpred (n)    (Zneg (Zsucc (Zneg n))))

(setq cZsum    (list 'Y (list '& 'f (list '& '(m n) (list (Zzero? 'n) 'm
                                  (list (Zminus? 'n) (Zpred (list 'f 'm (Zsucc 'n)))
                                   (Zsucc (list 'f 'm (Zpred 'n)))))))))

(defun Zsum (m n)    (list cZsum m n))

(defun Zprod (m n)    (list (Equiv (Zneg m) (Zneg n))
                            (Zplus (Nprod (Zmag m) (Zmag n)))
                            (Zminus (Nprod (Zmag m) (Zmag n)))))

(defun Qrat (x y)  (pair x y))

(setq 2/1    (Qrat p2 p1))

(setq 1/2    (Qrat p1 p2))

(setq 1/1   (Qrat p1 p1))

(defun Qnum (x)   (hd x))
```

```
(defun Qden (x)    (hd (tl x)))

(defun Qsum (x y)    (Qrat (Zsum (Zprod (Qnum x) (Qden y))
                                 (Zprod (Qden x) (Qnum y)))
                           (Zprod (Qden x) (Qden y))))

(defun Qprod (x y)
       (Qrat (Zprod (Qnum x) (Qnum y)) (Zprod (Qden x) (Qden y))))

(defun Qquo (x y)    (Qprod x (Qrecip x)))

(defun Qrecip (x)    (Qrat (Qden x) (Qnum x)))

(defun sqrt2  () (list 'Y (list '& 's (list '& 'k
                 (list (Neq n1 'k) 1/1
                  (Qprod 1/2 (Qsum (Qquo 2/1 (list 's (Npred 'k)))
                                   (list 's (Npred 'k)))))))))
```

Appendix 3: LISP Program to Compute Length of Seminormal Form of Square Root 2

```
(setq false   8)
(setq true   8)
(defun Not (x)   (add 2 x false true))
(defun Or (x y)   (add  2 x true y))
(defun Equiv (x y)   (add 2 x y (Not y)))
(setq 1of3   9)
(setq 2of3   9)
(setq 3of3   9)
(defun List (n h r)   (add 2 1 1 (add 2 1 n h r)))
(setq Nil   (List true  1 1))
(defun Cons (h r)   (List false h r))
(defun Null (x)   (add 2 x 1of3))
(defun hd (x)   (add 2 x 2of3))
(defun tl (x)   (add 2 x 3of3))
(defun pair (x y)   (Cons x (Cons y Nil)))
(setq n0   Nil)
(setq n1  (Cons Nil Nil))
(setq n2   (Cons Nil (Cons Nil Nil)))
(defun Nzero? (n)   (Null n))
(defun Nsucc (n)   (Cons Nil n))
(setq cNsucc (add 2 1 1 (Cons Nil 1)))
(defun Npred (x)   (tl x))
(setq rpt   (add 2 1 (add 2 1 1 (add 2 1 5
                    (add 2 (Nzero? 1) 1
                          (add 2 1 (add 2 1 1 (Npred 1) 1)))))))
(defun Nsum  (m)  (add 2 rpt cNsucc m))
(defun Nprod (m n)   (add 2 rpt (Nsum m) n n0))
```

```
(setq cNlt    (add 2 1 (add 2 1 1 (add 2 1 4 (add 2 (Nzero? 1) false
                          (add 2 (Nzero? 1) true
                            (add 2 1 (Npred 1) (Npred 1)))))))))

(defun Nlt (m n)    (add 2 cNlt m n))

(defun Ngt (m n)    (Nlt n m))

(defun Nne (x y)    (Or (Nlt x y) (Ngt x y)))

(defun Neq (x y)    (Not (Nne x y)))

(defun Zplus (n)    (pair false n))

(defun Zminus (n)    (pair true n))

(setq p1    (Zplus n1))

(setq p2    (Zplus n2))

(defun Zmag (n)    (hd (tl n)))

(defun Zminus? (n)    (hd n))

(defun Zplus? (n)    (Not (Zminus? n)))

(defun Zzero? (n)    (Nzero? (Zmag n)))

(defun Zneg (m)    (pair (Zplus? m) (Zmag m)))

(defun Zsucc (n)    (add 2 (Zzero? n) p1
                          (add 2 (Zplus? n) (Zplus (Nsucc (Zmag n)))
                                (Zminus (Npred (Zmag n))))))

(defun Zpred (n)    (Zneg (Zsucc (Zneg n))))

(setq cZsum    (add 2 1 (add 2 1 1 (add 2 1 4 (add 2 (Zzero? 1) 1
                          (add 2 (Zminus? 1) (Zpred (add 2 1 1 (Zsucc 1)))
                           (Zsucc (add 2 1 1 (Zpred 1)))))))))

(defun Zsum (m n)    (add 2 cZsum m n))

(defun Zprod (m n)    (add 2 (Equiv (Zneg m) (Zneg n))
                          (Zplus (Nprod (Zmag m) (Zmag n)))
                          (Zminus (Nprod (Zmag m) (Zmag n)))))

(defun Qrat (x y)  (pair x y))

(setq 2/1    (Qrat p2 p1))

(setq 1/2    (Qrat p1 p2))

(setq 1 1   (Qrat p1 p1))

(defun Qnum (x)   (hd x))
```

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Dudley Knox Library Code 0142 Naval Postgraduate School Monterey, CA 93943	3
Office of Research Administration Code 012A Naval Postgraduate School Monterey, CA 93943	1
Chairman, Code 52Ml Department of Computer Science Naval Postgraduate School Monterey, CA 93943	40
Bruce J. MacLennan Associate Professor and Chairman, Department of Computer Naval Postgraduate School Monterey, CA 93943	12
Dr. Robert Grafton Code 433 Office of Naval Research 800 N. Quincy Arlington, VA 22217	1
Dr. David Mizell Office of Naval Research 1030 East Green Street Pasadena, CA 91106	1

```
(defun Qden (x)    (hd (tl x)))

(defun Qsum (x y)    (Qrat (Zsum (Zprod (Qnum x) (Qden y))
                                 (Zprod (Qden x) (Qnum y)))
                           (Zprod (Qden x) (Qden y))))

(defun Qprod (x y)
       (Qrat (Zprod (Qnum x) (Qnum y)) (Zprod (Qden x) (Qden y))))

(defun Qquo (x y)    (Qprod x (Qrecip x)))

(defun Qrecip (x)    (Qrat (Qden x) (Qnum x)))

(defun sqrt2  () (add 2 1 (add 2 1 1 (add 2 1 1
                 (add 2 (Neq n1 1) 1/1
                  (Qprod 1/2 (Qsum (Qquo 2/1 (add 2 1 (Npred 1)))
                                   (add 2 1 (Npred 1)))))))))
```

END

FILMED

2-85

DTIC