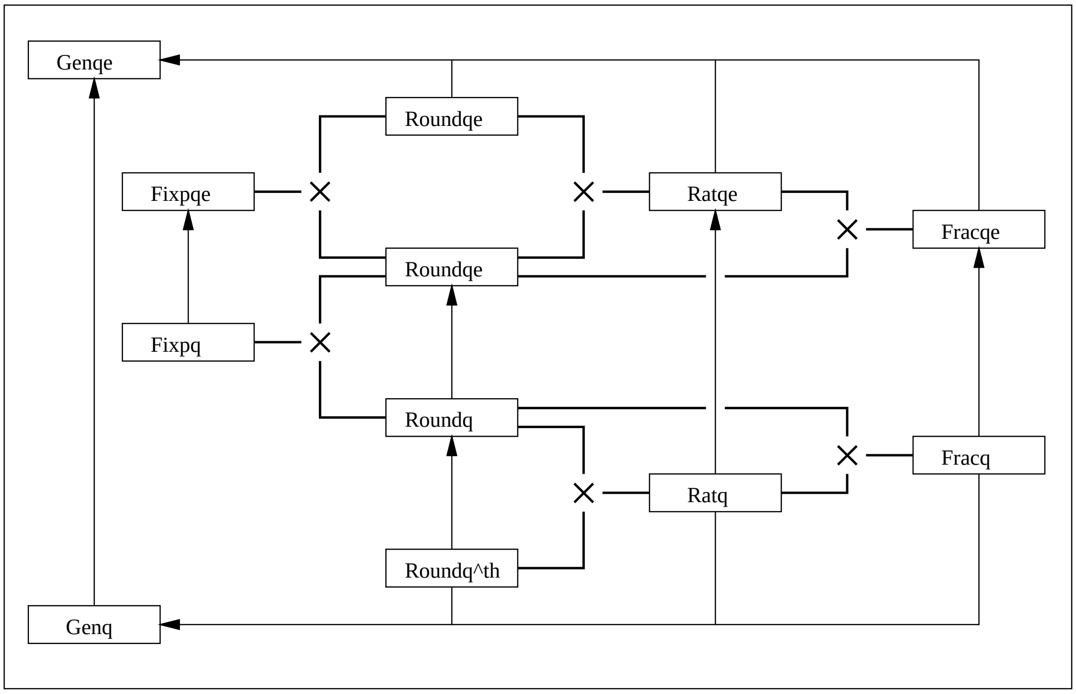
University of Amsterdam

Programming Research Group



An Arithmetical Module for Rationals and Reals

Jan Bergstra
Alex Sellink

Report P9709 June 1997

University of Amsterdam

Department of Computer Science

Programming Research Group

An arithmetical module for rationals and reals

Jan Bergstra
Alex Sellink

Report P9709 June 1997

J.A. Bergstra

Programming Research Group
Department of Computer Science
University of Amsterdam

Kruislaan 403
NL-1098 SJ Amsterdam
The Netherlands

tel. +31 20 525 7591
e-mail: janb@fwi.uva.nl

M.P.A. Sellink

Programming Research Group
Department of Computer Science
University of Amsterdam

Kruislaan 403
NL-1098 SJ Amsterdam
The Netherlands

tel. +31 20 525 7583
e-mail: alex@fwi.uva.nl

Universiteit van Amsterdam, 1997

An Arithmetical Module for Rationals and Reals

Jan Bergstra*,† Alex Sellink†

*Utrecht University, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

†University of Amsterdam, Programming Research Group
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

janb@wins.uva.nl alex@wins.uva.nl

1 Introduction

This paper is a sequel to our earlier paper [BS96]. We continue our project to develop a family of data type specifications starting from elementary principles and in a setting of four-valued logic. We refer to [BBR95] for a discussion on this particular four-valued logic, and to [BS96] for a discussion on its use.

The objective of this document is to provide a number of data types between naturals and rationals: natural numbers, fractions in various notations and fixed- and floating point reals in arbitrary precision. These fixed point reals result from rounding off which occurs if a decimal notation for a fraction is needed.

The specifications have been produced, typechecked and simulated using the ASF+SDF system [Kli95]. This system combines algebraic specifications [BHK85] with the Syntax Definition Formalism SDF [HK89]. We specify abstract syntax only. No module for pretty printing is provided. We notice that for full use in ASF+SDF the pretty printing options of ASF+SDF are not yet satisfactory. Equations like

$$\textit{monthly interest} = \textit{annual interest} \div\!\circ \; [1]\frown^{\circ}[2]$$

look a bit cumbersome. Preferably would be e.g.

$$\textit{monthly interest} = \frac{\textit{annual interest}}{12}$$

The rest of this paper is organised as follows. In Section 2 we introduce pairs and lists over arbitrary sorts. Then, in Section 6 we briefly discuss the three basic data types that play a role in this paper. For a detailed description of these data types we refer to [BS96]. Sections 8, 10, 12 and 14 contain descriptions of rounded quantities, rational quantities, fractioned quantities and their union respectively. Fixed point real quantities are introduced in section 18.

2 General Concepts

3 T4-0

imports Layout
exports
  sorts T4
  context-free syntax
    "D" → T4
    "T" → T4
    "F" → T4
    "M" → T4
    "eq1" "(" T4 "," T4 ")" → T4
    "(" T4 "◁" T4 "▷" T4 ")" → T4
    "(" T4 ")" → T4 {**bracket**}
  variables
    t[0-9′]∗ → T4
equations

The booleans form a four valued propositional algebra with constants D for the divergent value, T for true, F for false and M for meaningless, a conditional operator and an equality test. The intended rewriting strategy for the conditional operator is that it evaluates its second argument first, and applies one of the rules [1] – [4] thereafter. Variables t, t_i range over booleans (truth values).

[condD-T4] $(t_1 \lhd \mathrm{D} \rhd t_2) = \mathrm{D}$
[condT-T4] $(t_1 \lhd \mathrm{T} \rhd t_2) = t_1$
[condF-T4] $(t_1 \lhd \mathrm{F} \rhd t_2) = t_2$
[condM-T4] $(t_1 \lhd \mathrm{M} \rhd t_2) = \mathrm{M}$

The equations above are not specific for the booleans. For each sort we asume the presence of a conditional operator and the corresponding set of equations, i.e.

4 X4-0

imports T4-0[3]
exports
  sorts X4
  context-free syntax
    "D" → X4
    "M" → X4
    "(" X4 "◁" T4 "▷" X4 ")" → X4
    "eq2" "(" X4 "," X4 ")" → T4
    "T" "(" X4 ")" → T4
    "(" X4 ")" → X4 {**bracket**}
  variables
    x[0-9′]∗ → X4
equations

[condD-X4] $(x_1 \lhd \mathrm{D} \rhd x_2) = \mathrm{D}$

[condT-X4] $(x_1 \lhd \mathrm{T} \rhd x_2) = x_1$
[condF-X4] $(x_1 \lhd \mathrm{F} \rhd x_2) = x_2$
[condM-X4] $(x_1 \lhd \mathrm{M} \rhd x_2) = \mathrm{M}$

where x_1 and x_2 range over an arbitrary sort X. In the sequel we let x, x', x_i and y, y', y_i range over sorts X and Y respectively.

Contrary to the conditional operator — where we consider only the variant above that evaluates its second argument first — all binary operators in this document come in two versions, corresponding to left first- and right first evaluation of its arguments. Typically, eq1(.., ..) and ..${}_\circ\!\wedge$.. denote left first operators whereas eq2(.., ..) and ..$\wedge_\circ$.. denote right first operators. The right first version func2 of an arbitrary operator func can straightforwardly be obtained from the left first version func1 via

$$\mathrm{func2}(x, y) = (\mathrm{func1}(x, y) \lhd \mathrm{T}(y) \rhd \mathrm{M})$$

Analogously, the left first version can be obtained from the right first version. In case of a symmetrical operator func we can also use the identity $\mathrm{func1}(x, y) = \mathrm{func2}(y, x)$. In the sequel, we will not spell out both versions explicitly but restrict to giving one version if its counterpart is defined by one of the concepts above.

4.1 Pairs

In order to form terms that are in the cartesian product of two sorts (pairs) we have in-fix pairing operators ${}_\circ\!\bowtie$ and $\bowtie_\circ$. We let $\nwarrow xy$ denote the first projection and $xy\nearrow$ denote the second projection of xy. Furthermore we have a "true predicate" T which returns true unless the input pair or one of its components equals D or M.

5 XY4-1

imports X4-0[4] Y4-0 XY4-0 T4-1
exports
  context-free syntax
    X4 "$\bowtie_\circ$" Y4 $\to$ XY4
    X4 "${}_\circ\!\bowtie$" Y4 $\to$ XY4
    "$\nwarrow$" XY4 $\to$ X4
    XY4 "$\nearrow$" $\to$ Y4
equations

[lpr2-XY4]	$\nwarrow x \bowtie_\circ y = (x \lhd \mathrm{T}(y) \rhd \mathrm{M})$	[lpr1-X4]	$\nwarrow x \,{}_\circ\!\bowtie\, y = (x \lhd \mathrm{T}(x) \rhd \mathrm{M})$
[rpr2-XY4]	$x \bowtie_\circ y \nearrow = (y \lhd \mathrm{T}(y) \rhd \mathrm{M})$	[rpr1-X4]	$x \,{}_\circ\!\bowtie\, y \nearrow = (y \lhd \mathrm{T}(x) \rhd \mathrm{M})$

[lprM-XY4]	$\nwarrow \mathrm{M} = \mathrm{M}$	[lprD-XY4]	$\nwarrow \mathrm{D} = \mathrm{D}$
[rprM-XY4]	$\mathrm{M} \nearrow = \mathrm{M}$	[rprD-XY4]	$\mathrm{D} \nearrow = \mathrm{D}$

[lTrue-XY4] $\mathrm{T}(x \bowtie_\circ y) = \mathrm{T}(x) \wedge_\circ \mathrm{T}(y)$

[rTrue-XY4] $\mathrm{T}(x \,{}_{\circ}\!\!\ltimes\, y) = \mathrm{T}(x) \,{}_{\circ}\!\!\wedge\, \mathrm{T}(y)$
[TrueM-XY4] $\mathrm{T}(\mathrm{M}) = \mathrm{M}$
[TrueD-XY4] $\mathrm{T}(\mathrm{D}) = \mathrm{D}$

[M2-XY4]	$x \rtimes_{\circ} \mathrm{M} = \mathrm{M}$	[M1-XY4]	$\mathrm{M} \,{}_{\circ}\!\!\ltimes\, y = \mathrm{M}$
[D2-XY4]	$x \rtimes_{\circ} \mathrm{D} = \mathrm{D}$	[D1-XY4]	$\mathrm{D} \,{}_{\circ}\!\!\ltimes\, y = \mathrm{D}$

5.1 Lists

Lists over an arbitrary sort are definded by means of [ ] which maps x to the list $[x]$ of length 1. We use [ ] for the empty list and ${}^{\circ}\!\frown$ for (left first) concatenation. Again we have a "true predicate" T which returns true unless the input list or one of its members equals D or M. Then, we have a predicate E (empty) which tests whether the input list is empty and the head- and tail- operators on both sides of the list (⅂, ↱, ↳ and ↲). The equations for all these operators are given in [BS96].

6 Basic Sorts

The cartesian product and list operators defined in the previous section are defined for arbitrary sorts. In this section we define the three concrete sorts that play a role in this paper, namely booleans, signs and digits.

6.1 Booleans

Apart from the four conditional construct equations [1] – [4], we have equations for negation ($\neg$), left and right first conjunction and disjunction (${}_{\circ}\!\wedge$, $\wedge_{\circ}$, ${}^{\circ}\!\vee$ and $\vee^{\circ}$) the "true predicate" (T) and left first equality (eq1). Again, the equations can be found in [BS96].

6.2 Signs

7 S4-1

Let s, s', s_i range over signs. The sort of signs consists of $+$, $-$, the meaningless object M and the divergent object D.
imports S4-0 T4-1
exports
  context-free syntax
    S4 "$*_{\circ}$" S4 → S4 {**left**}
    "+" → S4
    "−" → S4
    "pos" "(" S4 ")" → T4
    "neg" "(" S4 ")" → T4
equations

[pos1-S4] $\mathrm{pos}(\mathrm{M}) = \mathrm{M}$

[pos2-S4] pos(D) $=$ D
[pos3-S4] pos(+) $=$ T
[pos4-S4] pos(−) $=$ F

[neg1-S4] neg(s) $= \neg$ pos(s)

[signprod-S4] $s_1 \ast s_2 = (s_1 \lhd \mathrm{pos}(s_2) \rhd (- \lhd \mathrm{pos}(s_1) \rhd +))$

[eq2-S4] eq2(s_1, s_2) $=$ pos($s_1 \ast_\circ s_2$)
[True-S4] T(s) $=$ eq2(s, s)

7.1 Digits

The sort of digits consists of the symbols $0, \ldots, 9$ and M and D. On digits we a successor function (S), a predecessor function (P) and a zero test (Z). Note that both S(9) and P(0) evaluate to M. Furthermore we have (as usual) a true predicate and an equality function. Finally we have a collection of elementary operators that are used to define arithmetical functions. Variables d, d', d_i range over digits.

8 Rounded Quantities

8.1 Rounded Quantity Expressions

The sort roundqe of rounded quantity expressions consists of lists of digits. Variables D, D', D_i range over roundqe.

8.2 Rounded Quanties

The sort roundq of rounded quantities consists of roundqe-terms without leading zeros. In particular, zero itself does not have leading zeros and is hence seen as the empty list of digits. We introduce a function rlz that removes leading zeros. The elementairy operations S (successor) and P (predecessor) as well as the predicate Z (being zero) can straightforwardly be expressed by means of the list operations of Section 2.

Apart from the arithmetical operations defined in [BS96] we need an operator ggd2 that computes the greatest common divisor of its arguments. The operator $\div_\circ$ computes the floor of the ratio of its arguments[1]. Variables n, n', n_i range over roundq.

9 N4-5

imports N4-4
exports
context-free syntax

[1] In [BS96] this operator is denoted as $\div_\circ$

“mod2” “(” N4 “,” N4 “)” $\rightarrow$ N4
“ggd2” “(” N4 “,” N4 “)” $\rightarrow$ N4

“upperbound” “(” N4 “)” $\rightarrow$ N4

“give-zeros” “(” N4 “)” $\rightarrow$ N4
“cut-off” “(” N4 “,” N4 “)” $\rightarrow$ N4

“min2” “(” N4 “,” N4 “)” $\rightarrow$ N4
“max2” “(” N4 “,” N4 “)” $\rightarrow$ N4
“dist2” “(” N4 “,” N4 “)” $\rightarrow$ N4

equations

[mod2-N4] $\mathrm{mod2}(n_1, n_2) = n_1 \multimap n_2 \times_{\circ} (n_1 \div_{\circ} n_2)$

[ggd2-N4] $\mathrm{ggd2}(n_1, n_2) = (n_1 \lhd \mathrm{Z}(n_2) \rhd \mathrm{ggd2}(n_2, \mathrm{mod2}(n_1, n_2)))$

Note that this definition of g.g.d. switches the arguments if the second argument is greater than the first argument. Note also that mod2([ ],n) reduces to [ ] and hence ggd2([ ],n) = ggd2(n,[ ]) = n for all positive round quantities n. This reflects the belief that each positive round quantity n is a divisor of [ ].

We end this module with a recursion equation which defines a function upperbound. This function returns the smallest power of the radix that is greater than the input value.

[ub-N4] $\mathrm{upperbound}(n) = ([1] \lhd \mathrm{Z}(n) \rhd \mathrm{upperbound}(\Lsh\, n) \overset{\frown}{\circ} [0])$

[min-N4] $\mathrm{min2}(n_1, n_2) = (n_1 \lhd n_1 <_{\circ} n_2 \rhd n_2)$
[max-N4] $\mathrm{max2}(n_1, n_2) = (n_2 \lhd n_1 <_{\circ} n_2 \rhd n_1)$

10 Rational Quantities

10.1 Rational Quantity Expressions

The sort ratqe of rational quantity expressions consists of pairs of roundqe-terms, i.e. ratqe = roundqe $\times$ roundqe. The rational quantity expression $n_1 \curlyvee_{\circ} n_2$ represents the *ratio* of n_1 and n_2, often denoted as $\frac{n_1}{n_2}$, n_1/n_2 or $n_1 : n_2$.

10.2 Rational Quanties

The sort ratq of rational quantities consists of pairs of ratq-terms with roundq components and with non-zero second component, i.e. ratq = roundq$\times$roundq$^+$, where roundq$^+$ is the sort containing all roundq-terms different from [ ]. We let q, q', q_i range over ratq.

11 Q4-2

imports Q4-1 N4-5[9]
exports
context-free syntax

"num" "(" Q4 ")"	$\rightarrow$ N4	
"den" "(" Q4 ")"	$\rightarrow$ N4	
"Z" "(" Q4 ")"	$\rightarrow$ T4	
"$\mathbb{S}$" "(" Q4 ")"	$\rightarrow$ Q4	
"inverse" "(" Q4 ")"	$\rightarrow$ Q4	
Q4 "$+\!\circ$" Q4	$\rightarrow$ Q4	{**left**}
Q4 "$\hat{+}\!\circ$" Q4	$\rightarrow$ Q4	{**left**}
Q4 "$-\!\circ$" Q4	$\rightarrow$ Q4	
Q4 "$\hat{-}\!\circ$" Q4	$\rightarrow$ Q4	
Q4 "$\times_\circ$" Q4	$\rightarrow$ Q4	{**left**}
Q4 "$\div\!\circ$" Q4	$\rightarrow$ Q4	{**left**}
Q4 "$<_\circ$" Q4	$\rightarrow$ T4	

equations

The numerator and denominator of q are the first and second projection of q.

[num-Q4] $\mathrm{num}(q) = \nwarrow q$
[den-Q4] $\mathrm{den}(q) = (\mathrm{M} \lhd \mathrm{Z}(q \nearrow) \rhd q \nearrow)$

Note that Z in the equation above is applied on a roundq-term. A ratq-term satisfies the predicate Z iff its numenator equals zero and its denominator is proper and unequal to zero, i.e. den(q) is proper.

[Z-Q4] $\mathrm{Z}(q) = \mathrm{Z}(\mathrm{num}(q)) \wedge_\circ \mathrm{T}(\mathrm{den}(q))$

ratq-terms of which the numerator and denominator have g.g.d. greater than 1 can be simplified.

[simpl-Q4] $$\frac{n = \mathrm{ggd2}(\mathrm{num}(q), \mathrm{den}(q))}{\mathbb{S}(q) = (\mathrm{num}(q) \div\!\circ\, n \rightthreetimes_\circ \mathrm{den}(q) \div\!\circ\, n \lhd [1] <_\circ n \rhd q)}$$

Note that the function simplify reduces $[\,] \rightthreetimes_\circ n$ to $[\,] \rightthreetimes_\circ [1]$ for all positive round quantity expressions n.

[inv1] $\mathrm{inverse}(q) = (\mathrm{den}(q) \rightthreetimes_\circ \mathrm{num}(q) \lhd \mathrm{T}(\mathrm{den}(q)) \rhd \mathrm{M})$

The product of two rational quantities is obtained by multiplying the numerators and the denominators. Division by q is multiplication with the inverse of q.

[mult-Q4] $q_1 \times_\circ q_2 = \mathbb{S}(\mathrm{num}(q_1) \times_\circ \mathrm{num}(q_2) \rightthreetimes_\circ \mathrm{den}(q_1) \times_\circ \mathrm{den}(q_2))$
[div-Q4] $q_1 \div\!\circ\, q_2 = q_1 \times_\circ \mathrm{inverse}(q_2)$

In order to define addition and subtraction we first define auxiliary addition and subtraction, meant to manipulate rational quantities with equal denominator.

[aux-add-Q4] $q_1 \hat{+}\!\circ\, q_2 = \mathrm{num}(q_1) +\!\circ\, \mathrm{num}(q_2) \rightthreetimes_\circ \mathrm{den}(q_1)$

[aux-sub-Q4] $q_1 \mathbin{\hat{-\!\circ}} q_2 = \mathrm{num}(q_1) \mathbin{-\!\circ} \mathrm{num}(q_2) \mathbin{\succ_\circ} \mathrm{den}(q_1)$

Now general addition and subtraction are defined as follows:

[add-Q4] $$\frac{n = \mathrm{den}(q_1) \times_\circ \mathrm{den}(q_2)}{q_1 \mathbin{+\!\circ} q_2 = \mathbb{S}(\mathrm{num}(q_1) \times_\circ \mathrm{den}(q_2) \succ_\circ n \mathbin{\hat{+\!\circ}} \mathrm{num}(q_2) \times_\circ \mathrm{den}(q_1) \succ_\circ n)}$$

[sub-Q4] $$\frac{n = \mathrm{den}(q_1) \times_\circ \mathrm{den}(q_2)}{q_1 \mathbin{-\!\circ} q_2 = \mathbb{S}(\mathrm{num}(q_1) \times_\circ \mathrm{den}(q_2) \succ_\circ n \mathbin{\hat{-\!\circ}} \mathrm{num}(q_2) \times_\circ \mathrm{den}(q_1) \succ_\circ n)}$$

[less-Q4] $q_1 <_\circ q_2 = \mathrm{num}(q_1) \times_\circ \mathrm{den}(q_2) <_\circ \mathrm{den}(q_1) \times_\circ \mathrm{num}(q_2)$

12 Fractioned Quantities

12.1 Fractioned Quantity Expressions

The sort fracqe of fractioned quantity expressions consists of pairs of roundqe and ratqe, i.e. fracqe = roundqe × ratqe. We let f, f', f_i range over fracq. The fractioned quantity expression $n \succ_\circ q$ represents the *fraction* consisting of n rounds and a ratio q, usually denoted by juxta position ($3\frac{4}{5}$, 3 4/5, ...).

12.2 Fractioned Quanties

The sort fracq of fractioned quantities consists of fracqe-terms with right component less than [1], i.e. fracq = roundq × ratq$^\theta$, where ratq$^\theta$ consists of all ratq-terms of which the left component is smaller than the right component.

13 F4-2

imports F4-1 Q4-2[11]
exports
  context-free syntax

"Z" "(" F4 ")"	→ T4	
"$\mathbb{S}$" "(" F4 ")"	→ F4	
"frac2rat" "(" F4 ")"	→ Q4	
"rat2frac" "(" Q4 ")"	→ F4	
F4 "$+\!\circ$" F4	→ F4	{**left**}
F4 "$-\!\circ$" F4	→ F4	{**left**}
F4 "$\times_\circ$" F4	→ F4	{**left**}
F4 "$\dot{\div}\!\circ$" F4	→ F4	{**left**}
F4 "$<_\circ$" F4	→ T4	

equations

The operator simplify shifts round quantities from the second to the first component of the fraction and then simplifies the second component:

[simplify] $\mathbb{S}(f) = \nwarrow f \mathbin{+\!\circ} \mathrm{num}(f\nearrow) \mathbin{\dot{\div}\!\circ} \mathrm{den}(f\nearrow) \succ_\circ \mathbb{S}(\mathrm{mod2}(\mathrm{num}(f\nearrow), \mathrm{den}(f\nearrow)) \succ_\circ \mathrm{den}(f\nearrow))$

The easiest way to define arithmetic operations on fractioned quantities is to convert to rational quantities, use the aritmetic operations on rational quantities and convert back. Therefore we define conversion operators frac2rat and rat2frac. On simplified terms frac2rat∘rat2frac and rat2frac∘frac2rat are identities.

[f2r] $\mathrm{frac2rat}(f) = \nwarrow f \times_{\circ} \mathrm{den}(f \nearrow) +_{\circ} \mathrm{num}(f \nearrow) \succ_{\circ} \mathrm{den}(f \nearrow)$

[r2f] $\mathrm{rat2frac}(q) = \mathbb{S}([\,] \succ_{\circ} q)$

[add-F4] $f_1 +_{\circ} f_2 = \mathrm{rat2frac}(\mathrm{frac2rat}(f_1) +_{\circ} \mathrm{frac2rat}(f_2))$
[sub-F4] $f_1 -_{\circ} f_2 = \mathrm{rat2frac}(\mathrm{frac2rat}(f_1) -_{\circ} \mathrm{frac2rat}(f_2))$
[prod-F4] $f_1 \times_{\circ} f_2 = \mathrm{rat2frac}(\mathrm{frac2rat}(f_1) \times_{\circ} \mathrm{frac2rat}(f_2))$
[div-F4] $f_1 \div_{\circ} f_2 = \mathrm{rat2frac}(\mathrm{frac2rat}(f_1) \div_{\circ} \mathrm{frac2rat}(f_2))$

[less-F4] $f_1 <_{\circ} f_2 = \mathrm{frac2rat}(f_1) <_{\circ} \mathrm{frac2rat}(f_2)$

14 General Quantities

14.1 General Quantity Expressions

In the previous sections we have defined arithmetic operations ($+_{\circ}$, $-_{\circ}$, $\times_{\circ}$, $\div_{\circ}$) on round quantities, rational quantities and fractioned quantities. The seperate definitions of the arithmetic operations on the different quantities is not quite satisfying. For instance the result of $\frac{1}{2} + \frac{5}{6}$ should be the fractioned quantity $1\frac{1}{3}$ rather than the rational quantity $\frac{4}{3}$. Moreover we want to be able to compute 'mixed' expressions like $\frac{3}{7} + 5\frac{4}{7}$ (giving 6). To this end we define — by means of three nameless injections — the sort genqe of general quantities as the disjoint union of round-, rational- and fractioned quantities:

$$
\begin{array}{lcl}
\mathrm{roundqe} & \rightarrow & \mathrm{genqe} \\
\mathrm{ratqe} & \rightarrow & \mathrm{genqe} \\
\mathrm{fracqe} & \rightarrow & \mathrm{genqe}
\end{array}
$$

14.2 General Quantities

The sort genq of general quantities is the disjoint union of the sorts roundq, ratq and fracq.

The operator Z (being zero) on general quantities is defined in terms of the same operator on the subsort in question. E.g. the general quantity n is zero iff the zero predicate defined on rounded quantities holds in n.

15 U4-3

imports U4-2
exports
context-free syntax

"$\mathbb{S}$" "(" U4 ")" $\to$ U4

"round2rat" "(" N4 ")" $\to$ Q4

U4 "$+_\circ$" U4 $\to$ U4
U4 "$-_\circ$" U4 $\to$ U4
U4 "$\times_\circ$" U4 $\to$ U4
U4 "$\div_\circ$" U4 $\to$ U4

U4 "$<_\circ$" U4 $\to$ T4

equations

Simplification of a rounded quantity comes down to removing the leading zeros.

[SimplU1] $\mathbb{S}(n) = \mathrm{rlz}(n)$

A fractioned quantity f is simplified by applying the simplification function of the fractioned quantities. If the rational (second) component equals zero, we take the rounded (first) component. Furthermore we write q instead of $[\ ]\rtimes_\circ q$ if q is unequal to zero. Note that the simplification function in the premise of rule **??** is defined on fractioned quantities, whereas the simplification function in the conclusion of **??** works on general quantities.

[SimplU3] $$\frac{f' \;=\; \mathbb{S}(f)}{\mathbb{S}(f) \;=\; (\nwarrow f' \lhd \mathrm{Z}(f' \nearrow) \rhd (f' \nearrow \lhd \mathrm{Z}(\nwarrow f') \rhd f'))}$$

Rational quantities are simplified by simplifying their conversion to a fractioned quantity:

[SimplU2] $\mathbb{S}(q) = \mathbb{S}(\mathrm{rat2frac}(q))$

Arithmetic operations on quantities in general are derived from the arithmetic operations on the rounded-, rational- and fractioned quantities by converting the operands to rationals. Note that conversion to fractions would not be efficient because computations with fractions are defined by means of conversion to rationals.

[N2R] $\mathrm{round2rat}(n) = n \rtimes_\circ [1]$

[SumU1] $n_1 +_\circ n_2 = \mathbb{S}(n_1 +_\circ n_2)$
[SumU2] $q_1 +_\circ q_2 = \mathbb{S}(q_1 +_\circ q_2)$
[SumU3] $f_1 +_\circ f_2 = \mathbb{S}(f_1 +_\circ f_2)$
[SumU4] $n +_\circ q = \mathbb{S}(\mathrm{round2rat}(n) +_\circ q)$
[SumU5] $n +_\circ f = \mathbb{S}(\mathrm{round2rat}(n) +_\circ \mathrm{frac2rat}(f))$
[SumU6] $q +_\circ f = \mathbb{S}(q +_\circ \mathrm{frac2rat}(f))$

[LessU1] $n_1 <_\circ n_2 = n_1 <_\circ n_2$
[LessU2] $q_1 <_\circ q_2 = q_1 <_\circ q_2$
[LessU3] $f_1 <_\circ f_2 = f_1 <_\circ f_2$
[LessU4] $n <_\circ q = \mathrm{round2rat}(n) <_\circ q$
[LessU5] $n <_\circ f = \mathrm{round2rat}(n) <_\circ \mathrm{frac2rat}(f)$
[LessU6] $q <_\circ f = q <_\circ \mathrm{frac2rat}(f)$

16 Addition of Signs

All arithmetical modules can be extended to their signed version by means of taking the cartesian product with the set of signs. In case of signed round quantities we get the following equations:

17 Z4-2

imports Z4-1 S4-1[7] N4-4
exports
  context-free syntax

"sign" "(" Z4 ")"	$\rightarrow$ S4
"abs" "(" Z4 ")"	$\rightarrow$ N4
"pos" "(" Z4 ")"	$\rightarrow$ T4
"neg" "(" Z4 ")"	$\rightarrow$ T4
"inv" "(" Z4 ")"	$\rightarrow$ Z4
"Z" "(" Z4 ")"	$\rightarrow$ T4
"S" "(" Z4 ")"	$\rightarrow$ Z4
"P" "(" Z4 ")"	$\rightarrow$ Z4
Z4 " $+\!\circ$ " Z4	$\rightarrow$ Z4
Z4 " $\multimap$ " Z4	$\rightarrow$ Z4
Z4 " $\times_{\circ}$ " Z4	$\rightarrow$ Z4
Z4 " $<_{\circ}$ " Z4	$\rightarrow$ T4
"min2" "(" Z4 "," Z4 ")"	$\rightarrow$ Z4
"max2" "(" Z4 "," Z4 ")"	$\rightarrow$ Z4

equations

[sign-Z4] $\mathrm{sign}(z) = \nwarrow z$

[abs-Z4] $\mathrm{abs}(z) = z \nearrow$

[inv-Z4] $\mathrm{inv}(z) = - *_{\circ}\ \mathrm{sign}(z) \times_{\circ} \mathrm{abs}(z)$

[succ-Z4] $\mathrm{S}(z) = (- \times_{\circ} \mathrm{P}(\mathrm{abs}(z)) \lhd \mathrm{neg}(z) \rhd + \times_{\circ} \mathrm{S}(\mathrm{abs}(z)))$

[pred-Z4] $\mathrm{P}(z) = \mathrm{inv}(\mathrm{S}(\mathrm{inv}(z)))$

On round quantities we have addition ($+\!\circ$) and subtraction ($\multimap$):

[add-Z4]
$$\frac{s_1 = \mathrm{sign}(z_1),\ s_2 = \mathrm{sign}(z_2)}{\begin{array}{l} z_1 +\!\circ\ z_2 = (z_2 +\!\circ\ z_1 \lhd \mathrm{abs}(z_1) <_{\circ} \mathrm{abs}(z_2) \\ \quad \rhd (s_1 \times_{\circ} \mathrm{abs}(z_1) +\!\circ\ \mathrm{abs}(z_2) \lhd \mathrm{pos}(s_1 *_{\circ} s_2) \\ \quad\quad \rhd s_1 \times_{\circ} \mathrm{abs}(z_1) \multimap \mathrm{abs}(z_2))) \end{array}}$$

[sub-Z4] $z_1 \multimap z_2 = z_1 +\!\circ\ \mathrm{inv}(z_2)$

Multiplication ($\times_o$):

[mult-Z4] $z_1 \times_o z_2 = \mathrm{sign}(z_1) *_o \mathrm{sign}(z_2) \rightarrowtail_o \mathrm{abs}(z_1) \times_o \mathrm{abs}(z_2)$

[zero-Z4] $\mathrm{Z}(z) = \mathrm{Z}(\mathrm{abs}(z))$

[pos-Z4] $\mathrm{pos}(z) = (\mathrm{F} \lhd \mathrm{Z}(z) \rhd \mathrm{pos}(\mathrm{sign}(z)))$
[neg-Z4] $\mathrm{neg}(z) = (\mathrm{F} \lhd \mathrm{Z}(z) \rhd \mathrm{neg}(\mathrm{sign}(z)))$

[less-Z4] $z_1 <_o z_2 = ((\mathrm{abs}(z_1) <_o \mathrm{abs}(z_2) \lhd \mathrm{pos}(z_1) \rhd \mathrm{T}) \lhd \mathrm{pos}(z_2)$
$\rhd (\mathrm{F} \lhd \mathrm{pos}(z_1) \rhd \mathrm{abs}(z_2) <_o \mathrm{abs}(z_1)))$

[Min-Z4] $\mathrm{min2}(z_1, z_2) = (z_1 \lhd z_1 <_o z_2 \rhd z_2)$
[Max-Z4] $\mathrm{max2}(z_1, z_2) = (z_2 \lhd z_1 <_o z_2 \rhd z_1)$

18 Fixed point Real Quantities

18.1 Fixed point Real Quantity Expressions

The sort fixprealqe of fixpoint real quantity expressions consists — similarly to ratqe — of pairs of roundqe, i.e. fixprealqe = roundqe × roundqe.

18.2 Fixed Point Real Quanties

The sort fixprealq of fixed point real quantities consists of fixprealqe-terms without leading zeros in the left component, i.e. fixprealq = roundq × roundqe. The intuition behind the pair $z \rightarrowtail_o D$ is not the fraction z/D but the decimal expression z, D. This explains why the second component is not a natural but a list of digits (e.g. $23,008 \neq 23,8$). We let r, r', r_i range over fixprealq.

19 R4-2

imports R4-1 N4-4
exports
context-free syntax
"aux-sum2" "(" R4 "," R4 "," N4 ")" → R4
"aux-sub2" "(" R4 "," R4 "," N4 ")" → R4
"step" "(" R4 ")" → R4
"pets" "(" R4 ")" → R4
"ten" "(" R4 ")" → R4
"net" "(" R4 ")" → R4
"tens1" "(" N4 "," R4 ")" → R4
"nets1" "(" N4 "," R4 ")" → R4
R4 "$+_o$" R4 → R4
R4 "$-_o$" R4 → R4
equations

In order to define addition we define auxiliary functions step, pets and nets1. The function step moves the decimal comma one position to the right. The function pets is the inverse of this operation.

[Step1]$\mathrm{step}(r) = \nwarrow r \ {}^{\circ}\!\frown \ [\Lsh r \nearrow] \rtimes_{\circ} \text{↳} r \nearrow$
[Pets1]$\mathrm{pets}(r) = \nwarrow r \ \text{↲} \rtimes_{\circ} [\nwarrow r \Rsh] \ {}^{\circ}\!\frown \ r \nearrow$

[Ten1]$\mathrm{ten}(r) = (\nwarrow r \frown\!{}^{\circ} [0] \rtimes_{\circ} [\,] \lhd \mathrm{E}(r \nearrow) \rhd \mathrm{step}(r))$
[Net1]$\mathrm{net}(r) = ([\,] \rtimes_{\circ} [0] \frown\!{}^{\circ} r \nearrow \lhd \mathrm{E}(\nwarrow r) \rhd \mathrm{pets}(r))$

The function nets1 repeatedly applies the function net. The number of repetitions is determined by the first argument of nets1. We also provide a function tens that repeated applies the function ten. We do not need function tens in this module but it used elsewhere.

[Nets1]$\mathrm{nets1}(n, r) = (r \lhd \mathrm{Z}(n) \rhd \mathrm{nets1}(\mathrm{P}(n), \mathrm{net}(r)))$
[Tens1]$\mathrm{tens1}(n, r) = (r \lhd \mathrm{Z}(n) \rhd \mathrm{tens1}(\mathrm{P}(n), \mathrm{ten}(r)))$

Addition ($+\!\!\circ$) also makes use of the auxiliary function aux-sum that shifts (if possible) the the decimal comma of its first two arguments one position to the right and administrates the number of shifts in its third argument. Afterwards, the comma is shifted back by means of the function shift1.

$$\begin{aligned}[\mathrm{Aux1}]\ \text{aux-sum2}(r_1, r_2, n) \ = \ & ((\mathrm{nets1}(n, \nwarrow r_1 \ +\!\!\circ \ \nwarrow r_2 \rtimes_{\circ} [\,]) \lhd \mathrm{E}(r_1 \nearrow) \rhd \mathrm{M}) \\ & \lhd \mathrm{E}(r_2 \nearrow) \\ & \rhd (\mathrm{M} \lhd \mathrm{E}(r_1 \nearrow) \rhd \text{aux-sum2}(\mathrm{ten}(r_1), \mathrm{ten}(r_2), \mathrm{S}(n))))\end{aligned}$$

$$\begin{aligned}[\mathrm{Aux2}]\ \text{aux-sub2}(r_1, r_2, n) \ = \ & ((\mathrm{nets1}(n, \nwarrow r_1 \ -\!\!\circ \ \nwarrow r_2 \rtimes_{\circ} [\,]) \lhd \mathrm{E}(r_1 \nearrow) \rhd \mathrm{M}) \\ & \lhd \mathrm{E}(r_2 \nearrow) \\ & \rhd (\mathrm{M} \lhd \mathrm{E}(r_1 \nearrow) \rhd \text{aux-sub2}(\mathrm{ten}(r_1), \mathrm{ten}(r_2), \mathrm{S}(n))))\end{aligned}$$

Now, addition and subtraction of fixpoint real quantity expressions is obtained by instantiating the third argument of the operators above with [ ]. Note that we do not consider additions and subtractions with different precision. That is to say, $r_1 +\!\!\circ r_2$ and $r_1 -\!\!\circ r_2$ reduce to M if the number of digits at the right of the decimal comma is different for r_1 and r_2. Expressions with different precisions are regarded as expressions of a different type.

[SumD]$r_1 \ +\!\!\circ \ r_2 = \text{aux-sum2}(r_1, r_2, [\,])$

[SubD]$r_1 \ -\!\!\circ \ r_2 = \text{aux-sub2}(r_1, r_2, [\,])$

20 R4-3

imports R4-2[19] F4-1 Q4-2[11]
exports
context-free syntax
"aux-mult-fl2" "(" R4 "," Q4 "," N4 ")" $\rightarrow$ R4

“aux-mult-cl2” “(” R4 “,” Q4 “,” N4 “)”	$\rightarrow$ R4
R4 “$\times_\circ$” Q4	$\rightarrow$ R4
N4 “$\ulcorner\!\div\!\circ$” N4	$\rightarrow$ N4
“mult-fl2” “(” R4 “,” Q4 “)”	$\rightarrow$ R4
“mult-cl2” “(” R4 “,” Q4 “)”	$\rightarrow$ R4
“fix2frac” “(” R4 “)”	$\rightarrow$ F4

equations

In order to build scalar multiplication on the sort of fixpoint reals we introduce an auxiliary function aux-mult-fl2 that shifts the comma to the right, computes the scalar multiplication and shifts the comma back. The scalar multiplication in aux-mult-fl2 comes down to multiplication with the numerator succeeded by division with the denominator. The division we use is the operator $\div\!\circ$ token from [BS96]. This division operator computes the floor of the ratio of its arguments.

Since we also want to have a scalar multiplication that rounds up instead of down, we also need a division that computes the ceiling of the ratio of its arguments. This latter operator is defined below and denoted as $\ulcorner\!\div\!\circ$. In order to emphasize that $\div\!\circ$ computes the floor of the ratio of its arguments we will denote this operator by $\llcorner\!\div\!\circ$ in the sequel.

The operator $\ulcorner\!\div\!\circ$ is easily obtained from $\llcorner\!\div\!\circ$ by adding the predessecor of denominator to the numerator:

[Div2] $n_1 \ulcorner\!\div\!\circ\ n_2 = (n_1 +\!\circ\ \mathrm{P}(n_2)) \llcorner\!\div\!\circ\ n_2$

The functions aux-mult-fl2 and aux-mult-cl2 are auxiliary functions for scalair multiplication using the floor-division and the ceiling-division respectively.

[Aux3] $\text{aux-mult-fl2}(r,\ q,\ n) \ = \ (\text{nets1}(n,\ (\nwarrow r \times_\circ \text{num}(q)) \llcorner\!\div\!\circ\ \text{den}(q) \rightarrowtail_\circ [\ ])$
$\lhd\ \mathrm{E}(r \nearrow)$
$\rhd\ \text{aux-mult-fl2}(\text{ten}(r),\ q,\ \mathrm{S}(n)))$

[Aux4] $\text{aux-mult-cl2}(r,\ q,\ n) \ = \ (\text{nets1}(n,\ \nwarrow r \times_\circ \text{num}(q) \ulcorner\!\div\!\circ\ \text{den}(q) \rightarrowtail_\circ [\ ])$
$\lhd\ \mathrm{E}(r \nearrow)$
$\rhd\ \text{aux-mult-cl2}(\text{ten}(r),\ q,\ \mathrm{S}(n)))$

Now, the following scalar multiplications compute the floor and the ceiling of the product of r and q.

[MultRQfl] $\text{mult-fl2}(r,\ q) = \text{aux-mult-fl2}(r,\ q,\ [\ ])$

[MultRQcl] $\text{mult-cl2}(r,\ q) = \text{aux-mult-cl2}(r,\ q,\ [\ ])$

Fixpoint real quantities are approximations. If the result of a computation is the fixpoint real quantity 6,25 this means that the result x of that computation satisfies $6,245 \leq x < 6,255$ rather than being exactly 6,25. In some cases, however, decimal notations refer to exact values. E.g. an interest of 6,25 % is usually exact. In those cases we should use fractions instead of fixpoint reals, i.e. $6\frac{25}{100}$ rather than 6,25. To this end we define a conversion function mapping fixpoint real quantities to fractioned quantities.

[Fix2frac] $\text{fix2frac}(r) = \nwarrow r \rightarrowtail_\circ r \nearrow \rightarrowtail_\circ \text{upperbound}(r \nearrow)$

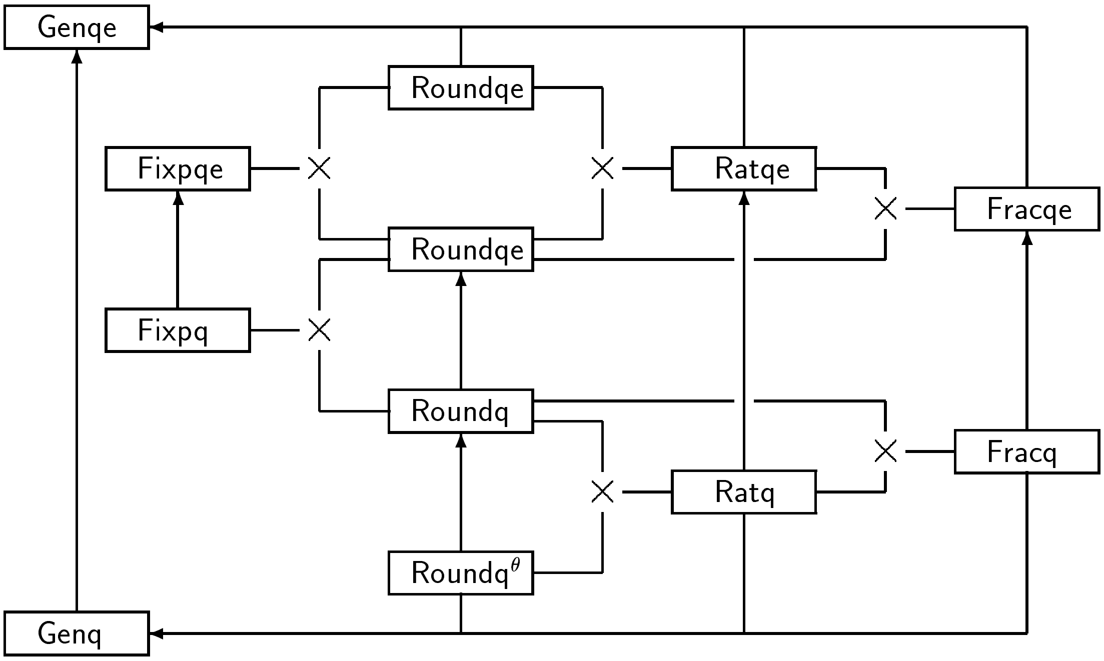


Figure 1: The cartesian- and the inclusion relations of the different sorts

21 Floating Point Real Quantities

21.1 Floating Point Real Quantity Expressions

The sort floprealqe of floating point real quantity expressions consists of pairs of which the components are itself pairs again: (digit × roundqe) × (sign × roundqe). The floating point real quantity expression $(d \curlyvee n) \curlyvee z$ represents the quantity with mantisse $d \curlyvee n$ and exponent z, usually denoted as $d, n \cdot 10^z$ or $d, n \times 10^z$ (i.e. $3,04 \cdot 10^{-5}$ or $3,04 \times 10^{-5}$ etc.)

21.2 Floating Point Real Quantities

The sort floprealq of floating point real quantities consist of floprealqe-terms of which the left component is a pair with a non-zero digit as its left component, and the right component is a signed round quantity, i.e. floprealq = (digit$^{\theta}$ × roundqe) × (sign × roundq). We let p, p', p_i range over floprealq.

22 P4-2

imports P4-1 D4-1 Z4-2[17] M4-1 R4-4
exports
  context-free syntax
    "mant" "(" P4 ")" → M4
    "exp" "(" P4 ")" → Z4
    "ten" "(" M4 ")" → M4
    "net" "(" M4 ")" → M4
    "𝕊" "(" P4 ")" → P4

“inc-exp” “(” P4 “)” $\rightarrow$ P4
“dec-exp” “(” P4 “)” $\rightarrow$ P4
“shift1” “(” N4 “,” P4 “)” $\rightarrow$ P4
“precision” “(” P4 “)” $\rightarrow$ Z4
“dec-prec2” “(” N4 “,” P4 “)” $\rightarrow$ P4

P4 “$\hat{+}\!\circ$” P4 $\rightarrow$ P4
P4 “$+\!\circ$” P4 $\rightarrow$ P4
P4 “$-\!\circ$” P4 $\rightarrow$ P4

“fix2float” “(” R4 “)” $\rightarrow$ P4
“float2fix” “(” P4 “)” $\rightarrow$ R4
“mant2fix” “(” M4 “)” $\rightarrow$ R4

equations

[Mant1] $\mathrm{mant}(p) = \nwarrow p$
[Exp1] $\mathrm{exp}(p) = p \nearrow$

The function ten multiplies the mantisse by radix. Note that only a matisse with left component equal to zero can succesfully be multiplied by radix. In case of a non-zero left component the result of multiplying by radix does not fit in the format of the mantisse. In those cases the result applying ten does not make sense.

[M-Ten1] $\mathrm{ten}(m) = \Lsh m \nearrow \rtimes_{\circ} \Rsh m \nearrow$

The function net divides the mantisse by radix. This is the inverse of the function ten that multiplies by radix.

[M-Net1] $\mathrm{net}(m) = 0 \rtimes_{\circ} [\nwarrow m] \frown^{\circ} m \nearrow$

The function inc-exp divides the mantisse by the radix and compensates this by incrementing the exponent by [1]. The function dec-exp works in the opposite direction. The intended use of the latter function is to apply it on floating point expressions starting with a zero digit.

[Inc-exp1] $\text{inc-exp}(p) = \mathrm{net}(\mathrm{mant}(p)) \rtimes_{\circ} \mathrm{S}(\mathrm{exp}(p))$
[Dec-exp1] $\text{dec-exp}(p) = \mathrm{ten}(\mathrm{mant}(p)) \rtimes_{\circ} \mathrm{P}(\mathrm{exp}(p))$

Simplification comes down to repeated application of the function dec-exp.

[SimplP4] $\mathbb{S}(p) = (\mathbb{S}(\text{dec-exp}(p)) \lhd \mathrm{Z}(\nwarrow \mathrm{mant}(p)) \rhd p)$

Computing by means of converting to fixed point reals does not work because different levels of precision in fixed point real computations results in meaningless. Therefore we use addition on round quantities. First we must make sure that the digits of which we compute the sum are of equal type in that sense that they both represent hundreds, thousands, etc.

[Prec] $\mathrm{precision}(p) = \mathrm{exp}(p) \multimap + \rtimes_{\circ} \mathrm{length}(\mathrm{mant}(p) \nearrow)$

In case of different precisions the computation of the sum the result should be rounded of according to the argument with lowest precision. This is equivalent to rounding of the highest precision before computing the sum. The latter strategy, however, is slightly simpler to implement.

$$
\text{[Dec-prec]}\quad
\dfrac{\begin{array}{rcl}
m & = & \mathrm{mant}(p), \\
n' & = & [\nwarrow m] \mathbin{\frown\!\circ} m \nearrow, \\
n'' & = & n' \mathbin{+\!\circ} [5], \\
p' & = & \Lsh\, n' \mathbin{\succ_{\circ}} \Rsh\, n' \lrcorner \mathbin{\succ_{\circ}} \exp(p), \\
p'' & = & \Lsh\, n'' \mathbin{\succ_{\circ}} \Rsh\, n'' \lrcorner \mathbin{\succ_{\circ}} \exp(p)
\end{array}}{\begin{array}{rcl}
\text{dec-prec2}(n,\, p) & = & (\mathrm{M} \lhd \mathrm{length}(m \nearrow) \mathbin{<_{\circ}} n \\
& & \rhd\, (p \lhd \mathrm{Z}(n) \\
& & \quad \rhd\, (p'' \lhd \mathrm{eq2}([1],\, n) \\
& & \qquad \rhd\, \text{dec-prec2}(\mathrm{P}(n),\, p'))))
\end{array}}
$$

$$
\text{[add-P4]}\quad
\dfrac{\begin{array}{rcl}
z & = & \mathrm{precision}(p_1) \multimap \mathrm{precision}(p_2), \\
p & = & \text{dec-prec2}(\mathrm{abs}(z),\, p_2)
\end{array}}{p_1 \mathbin{+\!\circ} p_2 \;=\; (p_2 \mathbin{+\!\circ} p_1 \lhd \mathrm{neg}(z) \rhd p_1 \mathbin{\hat{+}\!\circ} p)}
$$

$$
\text{[Aux-add-P4]}\quad
\dfrac{\begin{array}{rcl}
n_1 & = & [\nwarrow \mathrm{mant}(p_1)] \mathbin{\frown\!\circ} \mathrm{mant}(p_1) \nearrow, \\
n_2 & = & [\nwarrow \mathrm{mant}(p_2)] \mathbin{\frown\!\circ} \mathrm{mant}(p_2) \nearrow, \\
n & = & n_1 \mathbin{+\!\circ} n_2, \\
z & = & \mathrm{max2}(\exp(p_1),\, \exp(p_2)), \\
n' & = & \mathrm{length}(n) \multimap \mathrm{max2}(\mathrm{length}(n_1),\, \mathrm{length}(n_2)), \\
z' & = & z \mathbin{+\!\circ} + \mathbin{\succ_{\circ}} n'
\end{array}}{p_1 \mathbin{\hat{+}\!\circ} p_2 \;=\; \Lsh\, n \mathbin{\succ_{\circ}} \Rsh\, n \mathbin{\succ_{\circ}} z'}
$$

References

[BBR95] J.A. Bergstra, I. Bethke, and P. Rodenburg. A propositional logic with 4 values: true, false, divergent and meaningless. *J. Applied Non-Classical Logics*, 5(2):199–217, 1995.

[BHK85] J.A. Bergstra, J. Heering, and P. Klint. Algebraic definition of a simple programming language. Technical Report CS-R8504, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1985.

[BS96] J.A. Bergstra and M.P.A. Sellink. Sequential data algebra primitives. Technical Report P9602b, University of Amsterdam, Programming Research Group, 1996. revised version of P9602.

[HK89] J. Heering and P. Klint. The syntax definition formalism SDF. In J.A. Bergstra, J. Heering, and P. Klint, editors, *Algebraic Specification*, ACM Frontier Series, pages 283–296. Addison-Wesley, 1989.

[Kli95] P. Klint. *The ASF+SDF Meta-environment. User's Guide.* Centre for Mathematics and Computer Science, Amsterdam, 1995.

Technical Reports of the Programming Research Group

Note: These reports can be obtained using the technical reports overview on our WWW site (`http://www.fwi.uva.nl/research/prog/reports/`) or by anonymous ftp to `ftp.fwi.uva.nl`, directory `pub/programming-research/reports/`.

[P9709] J.A. Bergstra and M.P.A. Sellink. *An Arithmetical Module for Rationals and Reals.*

[P9705] M.G.J. van den Brand, M.P.A. Sellink, and C. Verhoef. *Generation of Components for Software Renovation Factories from Context-free Grammars.*

[P9704] P.A. Olivier. *Debugging Distributed Applications Using a Coordination Architecture.*

[P9703] H.P. Korver and M.P.A. Sellink. *A Formal Axiomatization for Alphabet Reasoning with Parametrized Processes.*

[P9702] M.G.J. van den Brand, M.P.A. Sellink, and C. Verhoef. *Reengineering COBOL Software Implies Specification of the Underlying Dialects.*

[P9701] E. Visser. *Polymorphic Syntax Definition.*

[P9618] M.G.J. van den Brand, P. Klint, and C. verhoef. *Re-engineering needs Generic Programming Language Technology.*

[P9617] P.I. Manuel. *ANSI Cobol III in SDF + an ASF Definition of a Y2K Tool.*

[P9616] P.H. Rodenburg. *A Complete System of Four-valued Logic.*

[P9615] S.P. Luttik and P.H. Rodenburg. *Transformations of Reduction Systems.*

[P9614] M.G.J. van den Brand, P. Klint, and C. Verhoef. *Core Technologies for System Renovation.*

[P9613] L. Moonen. *Data Flow Analysis for Reverse Engineering.*

[P9612] J.A. Hillebrand. *Transforming an ASF+SDF Specification into a ToolBus Application.*

[P9611] M.P.A. Sellink. *On the conservativity of Leibniz Equality.*

[P9610] T.B. Dinesh and S.M. Üsküdarlı. *Specifying input and output of visual languages.*

[P9609] T.B. Dinesh and S.M. Üsküdarlı. *The VAS formalism in VASE.*

[P9608] J.A. Hillebrand. *A small language for the specification of Grid Protocols.*

[P9607] J.J. Brunekreef. *A transformation tool for pure Prolog programs: the algebraic specification.*

[P9606] E. Visser. *Solving type equations in multi-level specifications (preliminary version).*

[P9605] P.R. D'Argenio and C. Verhoef. *A general conservative extension theorem in process algebras with inequalities.*

[P9602b] J.A. Bergstra and M.P.A. Sellink. *Sequential data algebra primitives (revised version of P9602).*

[P9604] E. Visser. *Multi-level specifications.*

[P9603] M.G.J. van den Brand, P. Klint, and C. Verhoef. *Reverse engineering and system renovation: an annotated bibliography.*

[P9602] J.A. Bergstra and M.P.A. Sellink. *Sequential data algebra primitives.*

[P9601] P.A. Olivier. *Embedded system simulation: testdriving the ToolBus.*

[P9512] J.J. Brunekreef. *TransLog, an interactive tool for transformation of logic programs.*

[P9511] J.A. Bergstra, J.A. Hillebrand, and A. Ponse. *Grid protocols based on synchronous communication: specification and correctness.*

[P9510] P.H. Rodenburg. *Termination and confluence in infinitary term rewriting.*

[P9509] J.A. Bergstra and Gh. Stefanescu. *Network algebra with demonic relation operators.*

[P9508] J.A. Bergstra, C.A. Middelburg, and Gh. Stefanescu. *Network algebra for synchronous and asynchronous dataflow.*

[P9507] E. Visser. *A case study in optimizing parsing schemata by disambiguation filters.*

[P9506] M.G.J. van den Brand and E. Visser. *Generation of formatters for context-free languages.*

[P9505] J.M.T. Romijn. *Automatic analysis of term rewriting systems: proving properties of term rewriting systems derived from* ASF+SDF *specifications.*

[P9504] M.G.J. van den Brand, A. van Deursen, T.B. Dinesh, J.F.Th. Kamperman, and E. Visser (editors). ASF+SDF*'95: a workshop on Generating Tools from Algebraic Specifications, May 11&12, 1995, CWI Amsterdam.*

[P9503] J.A. Bergstra and A. Ponse. *Frame-based process logics.*