Сборник задач по курсу Методы Формальных Спецификаций Программ: Язык функционального специфицирования RSL

Е.В. Корныхин

МГУ, 2007

Оглавление

Предисловие

Данный сборник задач предназначен студентам факультета вычислительной математики и кибернетики, изучающим курс методов формальных спецификаций программ. В нем представлены задачи по основным темам лекционных занятий, посвященных методологии RAISE и языку RSL, а также типовые задачи, предлагаемые студентам в виде зачетных работ по окончании прослушивания курса. Задачи первого типа применялись в преподавании семинарских занятий. Им посвящены главы 1-5. Задачи второго типа давались студентам на протяжении последних нескольких лет. Им посвящены остальные главы задачника. Наиболее трудные по мнению автора задачи помечены одной или двумя звездочками.

Каждая глава содержит небольшое теоретическое предисловие, не ставящее целью заменить лекционный материал, а лишь обозначить его основные точки. В главе 6 дано новое описание механизма параллельных взаимодействий в RSL, что обычно вызывает у студентов большие затруднения на зачетных работах. Материал теоретических предисловий остальных глав сознательно был сделан отличным от приведенных в методическом пособии [1]. Поэтому данный сборник задач и [1] следует использовать совместно. В теоретических предисловиях отмечен ряд особенностей языка RSL, на которые стоит обратить внимание студентам.

При изучении новой темы на семинаре желательно устное выполнение первых заданий (где нужно просто вычислить значение выражений). Это позволит студентам быстро понять некоторые особенности на простейших примерах. Обычно здесь сразу возникает много вопросов, почему ответ именно такой. Разъяснение этих моментов даст дополнительный толчок к правильному пониманию.

В конце данного сборника задач приведены две наиболее часто используемые таблицы при изучении языка RSL – таблица соответствия графического и текстового представления символов языка RSL и таблица приоритетов операций языка RSL.

Описание языка RSL можно прочитать в книге [4] RSL группы и ознакомиться с материалами курса в Университете МАКАО [3] и методическими пособиями, изданными на факультете ВМиК МГУ [2], [1]. При решении задач также можно пользоваться различными инструментами, проверяющими данные им спецификации на отсутствие синтаксических и семантических ошибок (type checker) [6], [5].

Глава 1

Скалярные типы RSL

Для написания предложений на языке RSL можно применять одну из двух нотаций: *графическую* или *текстовую*. Различие заключается в символах для записи операций и не-ASCII символов, привычных в математических текстах (буквах греческого алфавита). Полный список соответствия записи в графической и текстовой нотации можно посмотреть в Приложении А. Далее, если это специально не оговорено, будет использоваться графическая нотация (более употребительная в книгах об языке RSL).

1.1 Скалярные типы RSL

Встроенные типы языка RSL с операциями и отношениями перечислены в таблице 1.1.

chaos - константа, принадлежащая всем типам одновременно (аналог результата непредсказуемого поведения).

Условное выражение имеет привычный синтаксис: **if** e_1 **then** e_2 **else** e_3 **end** . e_2 и e_3 должны иметь одинаковый тип. Верны тождества:

$$\textbf{if true then } e_2 \textbf{ else } e_3 \textbf{ end } \equiv e_2$$

$$\textbf{if false then } e_2 \textbf{ else } e_3 \textbf{ end } \equiv e_3$$

$$\textbf{if chaos then } e_2 \textbf{ else } e_3 \textbf{ end } \equiv \textbf{chaos}$$

$$\textbf{if } e_1 \textbf{ then } e_2 \textbf{ else } e_3 \textbf{ end } \equiv \textbf{ if } e_1 \textbf{ then } e_2[\textbf{true}/e_1] \textbf{ else } e_3[\textbf{false}/e_1] \textbf{ end }, e_1 \sim \equiv \textbf{chaos}$$

$$\textbf{if } e_1 \textbf{ then } e_2 \textbf{ end } \equiv \textbf{ if } e_1 \textbf{ then } e_2 \textbf{ else skip end}$$

Синтаксис предикатов: $[\forall, \exists, \exists!]var : type \bullet boolexpr$. Обратите внимание, что символ $\bullet$ не является операцией!

Таблица 1.1: Встроенные типы языка RSL

название	значения	примеры	операции
Int	любое целое число (без ограничения величины)	1, 0-2	$+, -, \star, /, \backslash, \uparrow$, **abs**, **real**, $=, \neq, \equiv, >, <, \geqslant, \leqslant$
Nat	любое неотрицательное число (без наибольшего по величине числа)	1	$+, -, \star, /, \backslash, \uparrow$, **abs**, **real**, $=, \neq, \equiv, >, <, \geqslant, \leqslant$
Real	любое вещественное число бесконечной точности (без ограничения величины)	3.3, 0-4.5	$+, -, \star, /, \uparrow$, **abs**, **int**, $=, \neq, \equiv, >, <, \geqslant, \leqslant$
Bool	**true** и **false**	**true**	$\wedge, \vee, \sim, \Rightarrow, =, \neq, \equiv, \forall, \exists, \exists!$
Char	любой символ, представимый в цифровой-алфавитной системе	'a'	$=, \neq, \equiv$
Text	любая последовательность значений типа **Char**(список из **Char**)	"abba "56$%	см. главу 4, $=, \neq, \equiv$
Unit	**skip** (аналог void)		

1.2 Короткая логика в RSL

Операции-«логические связки» вычисляются согласно короткой логике. Остальные бинарные операции выполняются только после вычисления их аргументов.

Операция эквивалентности ($\equiv$) возвращает истину в том и только в том случае, если её аргументы суть одинаковые *«черные ящики»*, т.е. такие выражения, результаты вычислений которых будут одинаковыми (если прошли до конца, то одинаковые значения; если привели к **chaos**, то оба одновременно). Сравните: операция равенства ($=$) возвращает истину в том и только в том случае, если её аргументы *имеют одинаковое значение*. Операцию эквивалентности можно воспринимать, как безопасное сравнение на равенство.

Таблицы истинности логических операций смотрите в [1].

1.3 Задачи

I Вычислите значения следующих RSL-выражений

1. $2 + 3$
2. $3 + 2$
3. **abs** 2
4. **abs**$(0 - 2)$
5. **real** 2
6. **int** 2
7. **int** -2
8. **int**$(0 - 2)$
9. **int**$(2/4)$
10. **int**$(2 \setminus 4)$
11. **real**$(2 \setminus 4)$
12. $(2.0 \uparrow 0.5) \uparrow 2.0$
13. $(2 \uparrow 2) \uparrow 0.5$
14. $2 \uparrow (2.0 \uparrow 0.5)$
15. **real** $2 \uparrow (0.5 \uparrow 2.0)$
16. $'a' \neq 'b'$
17. $"aa" \equiv "bbb"$
18. **true** $\wedge$ **false**

II Упростить следующие выражения (сделать тождественное преобразование)

1. $x - x$ (x : **Int**)
2. $x \;/\; x$ (x : **Int**)
3. $x \setminus x$ (x : **Int**)
4. **if true then false else chaos end**
5. **if** a **then** $\sim (a \equiv$ **chaos**$)$ **else false end**
6. $x +$ **abs** $x = 0$
7. $x +$ **abs**$(0 - x) = 0$
8. **abs** $x +$ **abs**$(0 - x) = 0$
9. $x \wedge$ **true**
10. **true** $\wedge\, x$
11. $x \vee$ **true**
12. $x \Rightarrow$ **true**
13. $x \Rightarrow$ **false**
14. $x \Rightarrow x$
15. $x \Rightarrow x =$ **true**
16. $x \Rightarrow x \equiv$ **true**
17. $x \Rightarrow x \equiv x$
18. $x \Rightarrow x \Rightarrow x$
19. $x \Rightarrow y \Rightarrow x$

III Напишите на RSL логическое выражение, которое верно в том случае, если

1. n является количеством дней недели
2. n является количеством дней в июле
3. n является количеством дней в феврале
4. n чётно
5. n нечётно
6. n является составным числом
7. n является простым числом
8. n является степенью двойки

IV Какие из приведённых ниже тождеств верны в RSL:

1. $\sim\sim a \equiv a$
2. $\textbf{true} \vee a \equiv \textbf{true}$
3. $a \vee \textbf{true} \equiv \textbf{true}$
4. $a \Rightarrow \textbf{true} \equiv \textbf{true}$
5. $a \Rightarrow b \equiv (\sim a) \vee b$
6. $a \vee (\sim a) \equiv \textbf{true}$
7. $a \wedge (\sim a) \equiv \textbf{false}$
8. $(a \wedge b) \wedge c \equiv a \wedge (b \wedge c)$
9. $(a \vee b) \vee c \equiv a \vee (b \vee c)$
10. $(a = a) \equiv \textbf{true}$
11. $(a \equiv a) \equiv \textbf{true}$
12. $\forall\, x : \textbf{Nat} \bullet (x = 0) \vee (x > 0) \equiv (\forall\, x : \textbf{Nat} \bullet (x = 0)) \vee (x > 0)$

V Какие из следующих выражений верны

1. $\forall i : \textbf{Int} \bullet \exists j : \textbf{Int} \bullet i + j = 0$
2. $\forall i : \textbf{Int} \bullet \exists j : \textbf{Nat} \bullet i + j = 0$
3. $\exists i : \textbf{Int} \bullet \forall j : \textbf{Int} \bullet i + j = 0$
4. $\forall i : \textbf{Int} \bullet \exists! j : \textbf{Int} \bullet i + j = 0$
5. $\forall i, j : \textbf{Int} \bullet \exists! k : \textbf{Int} \bullet i + j = k$

VI Разное

1. Напишите на RSL выражение, выражающее тот факт, что нет наибольшего целого числа.
2. * Напишите на RSL выражение, выражающее тот факт, что n четно без использования мультипликативных операций
3. Запишите константу типа **Text**, состоящую из двойной кавычки.

Глава 2

Определение новых типов. Функции в RSL

2.1 Декартово произведение. Ограничение типа

В RSL допустимы следующие *типовые выражения* (способы описания новых типов)[1]:

1. встроенный тип
2. подтип (ограничение типа)
3. декартово произведение
4. множество
5. список
6. отображение

Подтип: $\{|var : typeexpr \bullet boolexpr|\}$
Декартово произведение: $typeexpr_1 \times typeexpr_2 \times ... \times typeexpr_n$

2.2 Способы задания функций

Явное задание функции:
$fname : typeexpr_1 \to typeexpr_2$
$fname(var) \equiv expr$

[1] типовые выражения обозначаются в схемах далее как $typeexpr$

Неявное задание функции:
$fname : typeexpr_1 \to typeexpr_2$
$fname(var_1)$ **as** var_2
post $expr$
Аксиоматическое задание функции:
$fname : typeexpr_1 \to typeexpr_2$
axiom $boolexpr$

У каждого такого способа задания функции может быть *предусловие*. Тогда функция называется *частично-вычислимой*.

2.3 Алгебраические спецификации

Операции над *целевым типом* могут быть описаны в виде алгебраической спецификации. Она состоит только из сигнатур функций и раздела аксиом. Каждая операция в ней должна принадлежать одной из трех категорий:

1. *функция-«обсервер»* – та, которая не меняет состояние параметра целевого типа (например, подсчет количества элементов, если целевой тип - контейнер)
2. *функция-«генератор»*
3. *функция-«трансформер»* – та, которая может быть реализована с использованием только функций-«генераторов» (например, удаление элемента из контейнера, если есть операции добавления и построение пустого контейнера)

Тогда в алгебраической спецификации обязательно должны присутствовать аксиомы вида:

$$\forall\, var : typeexpr \bullet o(g(var)) \equiv ...$$

$$\forall\, var : typeexpr \bullet o(t(var)) \equiv ...$$

где $o(.)$ - очередной обсервер, $g(.)$ - очередной генератор, $t(.)$ - очередной трансформер.

2.4 Простейший формат RSL-спецификации

scheme $schemename =$ **class**
$body$ **end**

body состоит из разделов функций, типов, глобальных переменных и аксиом. В теле может быть несколько разделов одного такого вида. В начале каждого раздела ставится специальное ключевое слово: для раздела функций - **value**, типов - **type**, глобальных переменных - **variable**, аксиом - **axiom**. Элементы внутри раздела перечисляются через запятую. Разделитель разделов – пробел.

2.5 Задачи

В занятие входят все задачи из методического пособия + приведённые ниже. Не решенные на занятии задачи идут в домашнюю работу.

Обратите внимание, что иррациональные константы не могут появляться в явных (или, что то же самое, исполнимых) спецификациях (по причине того, что их нельзя вычислить и предъявить для использования за конечное время). Однако их можно определить неявным или аксиоматическим образом.

Также обратите внимание на разницу фраз «вычислить» и «определить» («дать определение»). Первая означает написание явной (исполнимой, вычислимой) спецификации, а вторая – неявной или аксиоматической (дающей определение, но не способ вычисления).

I Какие из следующих спецификаций набраны без ошибок?

Какие ошибки сделаны?

1. ```
   vale f Int -> Int
         (f(x) is x)
   ```

2. ```
   value ff -> Int
         f() = 0
   ```

3. ```
   value f: Int => Int
         f x == x + 1
   ```

4. ```
   value f: (Int -> Int) -> Int
         f(g) is g(0)
   ```

5. ```
   value f: (Int -> Int) -> Int
         f(g) is f(0)
   ```
   ```

6.
```
value f: Int -> Bool
      f(x) is f(x)
```

7.
```
value f: Int -> Bool
      f(x) is ~f(x)
```

8.
```
value f: Int -> Int
      1 is f(x)
```

9.
```
value f: Int -> Int
      g(x) is f(x)
```

10.
```
value f: Int -> (Int, Real)
      f(x) is 0 >< 1.0
```

11.
```
value f: Int -> Int
      f(0) is 0
```

12.
```
value f, g: Int -> Int
      f(x) is x, g(x) is x+ 1
```

13.
```
value f: Int -> Int f(x) is x,
      f: Int -> Int f(x) is x + 1
```

14.
```
value f: Int -> Int f(x) is x,
      f: Int -> Int f(x) is x
```

15.
```
value f: Int -> Int f(x) is 0,
      f: Text -> Int f(x) is 1
```

II Построить сигнатурную спецификацию для следующих систем

Примечание: сигнатурная спецификация содержит только сигнатуры функций и определения типов в sort-виде

Спецификация должна без ошибок проходить этап проверки на type checker'e.

1. Написать спецификацию интерфейса программы-калькулятора.

III Построить явную и неявную спецификации на RSL для следующих программ

1. Вычисление максимума трех чисел $m = max(x, y, z)$
2. Переставить два числа $(x, y) = swap(u, v)$
3. Сложить два комплексных числа, заданных в алгебраической форме $z = c\_add(x, y)$ Опишите тип «Комплексное число».
4. Умножить два комплексных числа, заданных в алгебраической форме $z = c\_mul(x, y)$ Опишите тип «Комплексное число».
5. Вычислить факториал натурального числа $n! = ft(n)$
6. Вычислить НОД двух натуральных чисел $k = HOD(x, y)$
7. Вычислить n'е число Фибоначчи $f = fib(n)$
8. Получить два делителя числа минимальной величины $(a, b) = del(n)$
9. Отсортировать три числа $(a, b, c) = sort3(x, y, z)$
10. * Вычислить квадратный корень числа $s = sqrt(x)$
    Указание: в явной спецификации добавить параметр $epsilon$ – абсолютную погрешность измерения
11. Вычислить сумму квадратов двух максимальных чисел из трёх данных $m = max2(x, y, z)$
12. У Маши есть два брата - x и y. Первому n лет, второму m. Как зовут старшего брата Маши? $s = older(b_1, b_2), b_i : Brother$. Опишите тип «Brother».
13. Вычислить случайное число в отрезке между заданными двумя числами $r = random(x, y)$
14. Вычислить сумму первых n натуральных чисел (числа считаются с единицы) $s = sum1(n)$
15. Вычислить сумму квадратов первых n натуральных чисел (числа считаются с единицы) $s = sum2(n)$
    - рекурсию использовать можно
    - а теперь попробуйте без рекурсии

16. Среди шести данных чисел вывести номер максимального числа $n = max6(x1, x2, x3, x4, x5, x6)$

17. Даны 2 точки p_1 и p_2, заданные своими координатами в плоской декартовой системе координат, и число – радиус R. Вычислить координаты точки, лежащей в круге радиуса R, на границе которого лежат заданные две точки : $p = inside(p_1, p_2, R), p_1, p_2 : Point$. Опишите тип «Point».

18. Даны 2 точки, заданные своими координатами в плоской декартовой системе координат. Вернуть ту точку, которая ближе к началу координат $p = nearer(p_1, p_2), p_1, p_2 : Point$. Опишите тип «Point».

19. Дано 6 цифр – номер автобусного билета, который Вам дал контролер, пока Вы ехали на занятие в Университет. Проверить, является ли он «счастливым». Билет называется счастливым, если

    (a) первые три цифры совпадают с последними тремя

    (b) сумма цифр есть полный квадрат

    Опишите тип «Номер билета».

20. Даны 3 цифры. Можно ли из этих цифр собрать простое число.

    - необходимо использовать все цифры
    - среди данных цифр нет одинаковых
    - среди данных цифр могут быть одинаковые

21. Определить, к какому классу принадлежит данный IP-адрес $c = classofip(i), i : IP, c : Char$. Опишите тип «IP-адрес».

22. Проверить, является ли данный IP-адрес «частным» (т.е. является 10.0.0.0/8 или 172.16.0.0/12 или 192.168.0.0/16) $b = islocal(i), i : IP, c : Char$. Опишите тип «IP-адрес».

IV Построить аксиоматическую спецификации на RSL для следующих программ и понятий

1. Написать определение максимума двух чисел

2. Дана функция-отношение $\varphi(x, y)$. Дать определение того, что

    (a) φ рефлексивно

(b) φ симметрично

(c) φ транзитивно

(d) φ является отношением эквивалентности

(e) φ является отношением частичного порядка

3. Дана функция-«числовая последовательность» $x(n)$. Дать на RSL определение следующих математических фактов:

   (a) $x(n)$ ограничена

   (b) $x(n)$ монотонно возрастает

   (c) $x(n)$ сходится

   (d) * числовой ряд $\sum\limits_{k=0}^{+\infty} x(k)$ сходится

4. Дана вещественнозначная функция $f(x)$. Дать на RSL определение следующих математических фактов:

   (a) $f(x)$ ограничена

   (b) $f(x)$ монотонно возрастает

   (c) $f(x)$ непрерывна на всей области определения

   (d) * $f(x)$ дифференцируема на всей области определения

5. Даны типы X и Y. Дано отображение[2] $\varphi : X \to Y$. Дать на RSL определение следующих математических фактов:

   (a) φ тождественно ($\varphi(x) = C$) (попробуйте не использовать квантор существования)

   (b) φ тотально вычислима

   (c) φ частично вычислима

   (d) φ инъективно (различным аргументах соответствуют различные значения функции)

   (e) φ сюръективно (у каждого числа из области значений есть аргумент, который этому значению соответствует)

   (f) φ биективно (инъективно и сюръективно одновременно)

   (g) φ самообратимо ($\varphi(x) = \varphi^{-1}(x)$)

   (h) φ идемпотентно ($\varphi(\varphi(x)) = \varphi(x)$)

[2] в математическом смысле

6. Написать алгебраическую спецификацию типа «Список» и операций

   (a) получение пустого списка

   (b) добавления в список

      - к голове списка
      - * за конечным элементом списка

   (c) получение головы списка

   (d) получение хвоста списка

   (e) проверка нахождения элемента в списке

   С использованием своего определения определите константный список из элементов 1, 2, 3.

7. Написать спецификацию функции, вычисляющей длину списка. Список определить согласно задаче 6.

8. Написать спецификацию функции, конкатенирующую два данных списка. Список определить согласно задаче 6.

9. Написать алгебраическую спецификацию типа «Стек» и операций

   (a) получение пустого стека

   (b) добавление элемента в стек

   (c) удаление элемента из стека

   (d) получение головы стека

10. Написать алгебраическую спецификацию типа «Очередь» и операций

    (a) получение пустой очереди

    (b) добавление элемента в очередь

    (c) удаление элемента из очереди

    (d) получение головы очереди

11. Написать алгебраическую спецификацию типа «Двоичное дерево» и операций

    (a) получение пустого двоичного дерева

    (b) добавление элемента в двоичное дерево

(c) проверка нахождения элемента в дереве

Уточнить тип «Двоичное дерево» до типа «Двоичное дерево поиска».

12. Написать определение натурального числа в виде схемы языка RSL, используя аксиоматику Пеано:

    - определите константу zero и функцию $succ$ (она вычисляет следующий элемент)
    - напишите определение следующих аксиом:
      (a) zero – минимальный элемент
      (b) «линейный порядок»: для равенства последователей элементов необходимо равенство самих элементов
      (c) «математическая индукция»

13. Пользуясь определением натурального числа из предыдущей задачи, дать определение операций сложения и умножения двух натуральных чисел.
    Примечание: Операции сложения и умножения надо натуральным числами можно описать следующими аксиомами:

    (a) $\forall x \in \mathbb{N}\ (+(x, 0) = x)$
    (b) $\forall x, y \in \mathbb{N}\ (+(x, succ(y)) = succ(+(x, y)))$
    (c) $\forall x \in \mathbb{N}\ (*(x, 0) = 0)$
    (d) $\forall x, y \in \mathbb{N}\ (*(x, succ(y)) = +(*(x, y), x))$

14. Написать определение предела заданной числовой последовательности $a(n)$ (например, «на языке $\varepsilon - \delta$»). Считайте, что числовая последовательность имеет следующее определение: a : **Nat** $\to$ **Real**.

15. Написать определение суммы заданного числового ряда, составленного из элементов числовой последовательности $a(n)$, используя определение предела из предыдущей задачи. Считайте, что числовая последовательность имеет следующее определение: a : **Nat** $\to$ **Real**. Считайте, что ряд считается с $n = 0$.

16. * Определите на RSL константу e (неявно)
    Примечание: Представьте, что кто-то вычислил вещественное число (e). Вам остаётся проверить, что оно действительно есть e. Например, для этого можно неявным образом использоваться опреде-

ление e из теории пределов или числовых рядов:
$e = \lim\limits_{k \to +\infty} (1 + \frac{1}{k})^k = \sum\limits_{k=0}^{+\infty} \frac{1}{k!}$

17. * Определите на RSL константу π (неявно)
Примечание: $\frac{\pi}{4} = \sum\limits_{k=0}^{+\infty} \frac{(-1)^k}{2k-1}$

18. * Дать определение синуса и косинуса числа $s = \sin(x)$, $t = \cos(x)$
Примечание: следующий набор аксиом однозначно определяет две функции $S(x)$ и $C(x)$, которые равны синусу и косинусу x соответственно :

$$\text{для любых } x, y : S(x+y) = S(x)C(y) + C(x)S(y)$$

$$\text{для любых } x, y : C(x+y) = C(x)C(y) - S(x)S(y)$$

$$\text{для любого } x : S^2(x) + C^2(x) = 1$$

$$S(\frac{\pi}{2}) = 1$$

$$\text{для любого } x \in (0, \frac{\pi}{2}) : 0 < S(x) < x$$

Обоснование этого факта приведено в книге Ильина, Позняка «Математический анализ», том 2
Примечание: считать константу π определенной (например, из предыдущей задачи)

19. ** Определите явным образом на RSL синус с заданной абсолютной погрешностью
Примечание: $\sin(x) = \sum\limits_{k=0}^{+\infty} \frac{(-x^2)^k}{(2k-1)!!}$

V Построить явную спецификацию по неявной

1.
```
value f: Nat -> Bool
f(n) as b
post
        if exists x:Nat :- exists y:Nat :- n = x + y
            /\ y = x then ~b else b
        end
```

2.
```
value f: Nat >< Nat -> Bool
f(x, y) as b
post
        if all n:Nat :- n - x = n - y \/
            (exists m:Text :- m ~= ""
            => (exists c, d: Nat :- c = 2 ** d))
            then ~b else b
        end
```

3.
```
value f: Nat -> Nat
f(n) as k
post
(
    (all i:Nat :- i >= 1 /\ i <= n => k \ i = 0)
    /\
    (all i:Nat :- i >= 1 /\ i < n /\ k \ i = 0 =>
        ( exists! j:Nat :- j >= 1 /\ j <= n /\ k \ j = 0
        /\ j > i )
    )
)
```

4.
```
value
    f: Nat -> Nat
    f(n) as k
    post g(k, n),

    g: Nat >< Nat -> Bool
    g(n, k) as b
    post
        if k = 0 then b
        elsif n < k then ~b
        else g(n-k, k-1)
        end
```

VI Спецификация автоматного поведения

Шаблон спецификации автоматного поведения :

typeСостояние, Вход, Выход

value

старт : Состояние,

переход: Состояние $\times$ Вход $\to$ Состояние $\times$ Выход

1. Напишите спецификацию трехцветного автомобильного светофора

2. Напишите спецификацию двухцветного пешеходного светофора

3. Напишите спецификацию ячейки динамической оперативной памяти (сигнал 0 – обновить значение, сигнал 1 – сохранить значение)

4. Некая фирма скоро начнёт производство автоматов по продаже шипучего напитка «Буратино». В автомате в каждый момент находится некоторое количество бутылок этого бодрящего напитка. Каждая бутылка стоит некоторое количество денежных единиц. На автомате есть кнопка с надписью «Нажми - если загорится зелёная лампочка, значит, бутылки есть; если красная, то бутылок в автомате сейчас нет». Цена бутылки «Буратино» написана на автомате. Для покупки напитка следует нажать кнопку «Пуск» и вставить деньги в соответствующее отверстие. После чего автомат выдаст максимальное количество бутылок «Буратино» на данные ему деньги, а также выдаст сдачу. В фирме будет специальный человек, который с некоторой периодичностью будет приезжать к автомату, загружать его бутылками «Буратино» и забирать из него деньги.

   Напишите спецификацию поведения такого автомата по продаже шипучего напитка «Буратино».

   (a) как изменится спецификация, когда после выполненной Вами работы выяснится, что разработчики автомата решили дополнить его такой возможностью: цена бутылки может меняться – менять её сможет тот же самый человек, который собирает деньги из автомата ?

5. За неудовлетворительную работу Московский метрополитен решает заменить всех женщин в кассах на автоматы. На автомате есть кнопки, соответствующие тому, на сколько поездок надо купить билет. Автомат должен принимать деньги, а взамен выдавать билеты и сдачу. К Вам обратились с просьбой написать спецификацию данного автомата, дабы пассажиры метрополитена не заметили разницы, а скорость продвижения очередей сократилась бы. Примите эту просьбу и напишите спецификацию поведения новых автоматов по продаже билетов в метро.

6. На железнодорожном вокзале решают ввести автоматическую систему, распределяющую пути по приезжающим поездам. Поезд может занимать путь на неопределённое время, но о том, занят ли

путь, всегда можно узнать. Количество путей на вокзале в системе не фиксируется и будет задано на этапе внедрения. Напишите спецификацию такой автоматической системы.

7. Напишите спецификацию банкомата. Банкомат при правильном вводе пин-кода может выдавать деньги, выдавать баланс. Есть человек, который с некоторой периодичностью приезжает к этому банкомату и кладёт туда деньги.

8. Напишите спецификацию поведения игроков в игре «Крестики-Нолики».
   Примечание: например, такая спецификация может содержать определения следующих функций: ход нолика, ход крестика, выиграл ли кто-либо сейчас?, пустое поле (это будет константа)

9. Напишите спецификацию поведения в игре «Виселица»: один игрок загадывает слово, а другой, называя буквы, пытается отгадать его. При этом количество попыток ограничено (правда, каким числом конкретно специфицировать пока не нужно). Шаг загадывания слова опишите в sort-виде.

10. Напишите спецификацию поведения в игре «Палочки». Правила игры такие. Перед двумя игроками лежит некоторое количество палочек. Поочерёдно они вытаскивают от одной до трёх палочек. Проигрывает тот, кто берёт последнюю палочку.

VII Предложить уточнение поведения

Дана алгебраическая спецификация. Предложить уточнение для неё в виде неявной или явной спецификаций.

1.
```
value f, g : Int -> Int
axiom all x: Int :- f(g(x)) is x
```

2.
```
value C: Int, f, g : Int -> Int
axiom all x: Int :- f(g(x)) is C
```

3.
```
value C: Int, f, g : Int -> Int
axiom all x: Int :- f(x) is g(C)
```

4.
```
value g : Int -> Int, f: Int -> Bool
axiom all x: Int :- f(g(x)) is false
```

5.
```
value f : Int -> Int
axiom all x: Int :- f(x * 2) is f(x) + f(x)
```

6.
```
value f : Int -> Int
axiom all x: Int :- f(x * 2) is f(x) * f(x)
```

7.
```
value f : Int -> Int
axiom all x, y: Int :- f(x * x + y * y) is f(x) + f(y)
```

8.
```
value f: Int >< Int -> Bool, g: Int -> Bool
axiom
    all x, y: Int :- f(x, y) is
        if g(y) then x = 1 else g(x) end,
    all x: Int :- g(x) is
        if f(x, 0) then x = 1 else g(x - 1) end
```

9.
```
value f: Int >< Int -> Bool, g: Int -> Bool
axiom
    all x, y: Int :- f(x, y) is
        if g(y) then x = 1 else g(0) end,
    all x: Int :- g(x) is
        if f(x, x) then x = 0 else g(x - 1) end
```

VIII Доказательство корректности уточнения

Доказать, что схема S2 является уточнением схемы S1

1.
```
scheme S1 = class
    type
        A, B, C
    value
        f1 : A >< B -> Bool,
        f2 : A >< B -~-> C,
```

```
        f3 : C -~-> A >< B
    axiom
        all a : A, b : B :-
            f3( f2(a,b) ) is (b, a)
        pre f1(a, b)
end


scheme S2 = class
    type
        A = Int,
        B = Int,
        C = Int
    value
        f1 : A >< B -> Bool
        f1(i, n) is i + 1 = n,

        f2 : A >< B -~-> C
        f2(i, n) is n,

        f3 : C -~-> A >< B
        f3(x) as (a, b)
        post x = a /\ b = a - 1
end
```

2.

```
scheme S1 = class
    type
        A, B, C
    value
        f1 : A >< B -> C,
        f2 : B >< A -~-> C,
        f3 : A -> B >< A,
        f4 : B -> A >< B
    axiom
        all a : A, b : B :-
            f1(f4(b)) = f2(f3(a))
        pre f1(a, b) = f2(b, a)
    end
```

```
scheme S2 = class
    type
        A = Int,
        B = Nat,
        C = Int
    value
        f1 : A >< B -> C
        f1(a, b) is a,

        f2 : B >< A -~-> C
        f2(a, b) is a,

        f3 : A -> B >< A
        f3(a) is (a, a),

        f4 : B -> A >< B
        f4(b) is (b, b)
    end
```

IX Разное

1. Студент Иванов написал такое определение на RSL того факта, что $\lim\limits_{n\to+\infty} \frac{n+1}{n} = 1$

   ```
   all `epsilon:Real :- `epsilon > 0 =>
   exists N: Nat :-
   all n > N =>
       abs( (n+1)/n ) - 1 ) < `epsilon
   ```

   Помогите Иванову получить зачёт по МФСП, т.е. сделать так, чтобы RSL TYPE CHECKER на его спецификации не выдавал сообщений об ошибках

2. Какие из приведенных ниже выражений являются правильной спецификацией константы:

   (a) `value x = 0`

   (b) `value x :- 0`

   (c) `value x: Int :- 0`

(d) `value x: Int :- x = 0`

(e) `value x: Int :- true`

(f) `value x: Int :- x = 0 \/ x = 1`

(g) `value x: Int :- x = 0 /\ x = 1`

3. Упростить следующие выражения:

(a)
```
value
       f: Int >< Int >< Int -> Bool
       f(x, y, z) is if let x = y then z = y end
           then true else false end
```

(b)
```
value
       f: Int >< Int >< Int -> Int
       f(x, y, z) is
           let (x, z) = (x + z, y - x) in
           let (x, y) = (y - z, x) in
               x + y + z
           end end
```

Глава 3

Множества

3.1 Определения и операции

Множество – это контейнер элементов одного типа, который обладает свойствами уникальности и неупорядоченности элементов.

Типовое выражение для конечного множества: $typeexpr$**-set**. Типовое выражение для бесконечного множества: $typeexpr$**-infset**. Конечное множество – подтип бесконечного множества.

Операции:

$=$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – сравнение на равенство

$\neq$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – сравнение на не равенство

$\cup$: T**-infset** $\times$ T**-infset** $\rightarrow$ T**-infset** – объединение множеств

$\cap$: T**-infset** $\times$ T**-infset** $\rightarrow$ T**-infset** – пересечение множеств

$\setminus$: T**-infset** $\times$ T**-infset** $\rightarrow$ T**-infset** – вычитание множеств

$\in$: $T \times T$**-infset** $\rightarrow$ **Bool** – проверка на принадлежность

$\notin$: $T \times T$**-infset** $\rightarrow$ **Bool** – проверка на непринадлежность

$\subset$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – проверка вложения

$\supset$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – проверка вложения

$\subseteq$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – проверка вложения

$\supseteq$: T**-infset** $\times$ T**-infset** $\rightarrow$ **Bool** – проверка вложения

card : T**-infset**$\widetilde{\rightarrow}$**Nat** – количество элементов (**chaos** для бесконечных множеств)

Конструктор множества:

(*пустое множество*) {}

(*перечисление*) {0, 1, 2} - множество, состоящее из трех целых чисел (не натуральных!) - нуля, единицы и двойки.

(*диапазон*) {0..2} - то же, что и {0, 1, 2}; {0..0} ≡ {0}, {1..0} ≡ {}

(*«сокращенная запись»*) $\{expr\_with\_var | var : typeexpr \bullet boolexpr\}$ (например, $\{2 \star n | n : \mathbf{Nat} \bullet n < 3\}$, что эквивалентно {0, 2, 4})

3.2 Задачи

В занятие входят все задачи из методического пособия ([1]) + приведённые ниже. Не решенные на занятии задачи идут в домашнюю работу.

I Вычислить

1. $\{1, 2\} = \{3, 1\}$
2. $\{1, 2\} = \{2, 1\}$
3. $\{1, 2, 1\} = \{2, 2, 1\}$
4. $\{1, 2\} \cup \{3, 4\}$
5. $\{1, 2\} \cup \{2, 3\}$
6. $\{1, 2\} \cap \{3, 4\}$
7. $\{1, 2\} \cap \{2, 3\}$
8. $\{1..30\} \cup \{10.. - 10\}$
9. $\{1..30\} \cap \{10.. - 10\}$
10. $\{1..30\} \cup \{x | x : \mathbf{Int} \bullet \mathbf{abs}\, x < 11\}$
11. $\{1..30\} \cap \{x | x : \mathbf{Int} \bullet \mathbf{abs}\, x < 11\}$
12. $\{x + 10 | x : \mathbf{Int}\} = \{x | x : \mathbf{Int}\}$

13. $\{5 * k + 2 | k : \mathbf{Int}\} \cap \{3 * k - 1 | k : \mathbf{Int}\}$
14. $\{5 * k + 2 | k : \mathbf{Int}\} \setminus \{3 * k - 1 | k : \mathbf{Int}\}$
15. $\{5 * k + 2 | k : \mathbf{Int}\} \subset \{3 * k - 1 | k : \mathbf{Int}\}$
16. $\{5 * k + 2 | k : \mathbf{Int}\} \supseteq \{3 * k - 1 | k : \mathbf{Int}\}$
17. $\sim (\{1, 2\} \subseteq \{2, 1, 1\})$
18. $\mathbf{card}\{1..30\}$
19. $\mathbf{card}\{5 * k + 2 | k : \mathbf{Int} \bullet k * k \in \{-10..10\}\}$
20. $\mathbf{card}\{5 * k + 2 | k : \mathbf{Int} \bullet k * k \in \{10.. - 10\}\}$
21. $\mathbf{card}\{k * k - 2 * k | k : \mathbf{Int} \bullet k * k \in \{-10..10\}\}$
22. $\forall x : \mathbf{Nat} \bullet x \in \{x | x : \mathbf{Int}\}$
23. $\forall x : \mathbf{Int} \bullet x \in \{x | x : \mathbf{Nat}\}$

II Решите уравнения

1. $\{1\} \cup x = \{\}$
2. $\{1\} \cup x = \{1\}$
3. $\{1\} \cup x = \{1, 2\}$
4. $\{1\} \cap x = \{\}$
5. $\{1\} \cap x = \{1\}$
6. $\{1\} \cap x = \{1, 2\}$
7. $\{2, 1\} \setminus x = \{1\}$
8. $\mathbf{card}\, x = 0$
9. $\mathbf{card}\, x = 1$

III Какие их следующих выражений верны

Считайте, что свободные переменные располагаются под квантором всеобщности

1. $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$
2. $(A \setminus B) \cap C = (A \cap C) \setminus B$
3. $(A \cup B) \setminus C \equiv (A \setminus C) \cup (B \setminus C)$
4. $A \cup \{\} = A$
5. $\{\} \cup A = A$
6. $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
7. **card Nat** $<$ **card Int**
8. **card Nat** $=$ **card Int**
9. **card Nat** $>$ **card Int**
10. **card Nat** $\equiv$ **card Int**
11. **card**$\{n \mid n : \textbf{Nat}\} \equiv$ **card**$\{n \mid n : \textbf{Int}\}$

IV Записать на RSL следующие константы и определения типов

1. Пустое множество
2. Множество чисел 1, 2, 3 (а также множество чисел от 1 до 3)
3. Множество всех чётных чисел
4. Множество всех чётных чисел, не превышающих 10 (привести в виде перечисления и нескольких различных сокращённых формах)
5. Множество всех простых натуральных чисел
6. Множество всех пар взаимнопростых натуральных чисел
7. Множество всех троек, в каждой из которых есть одинаковые элементы

8. Множество всех степеней двойки, не превышающих 100 (привести в виде перечисления и нескольких сокращённых формах)
9. Множество всех IP-адресов класса A (B, C, D, E)
10. Множество всех точек плоскости, образующих
    (a) Прямую – биссектрису I и III квадрантов
    (b) Правую полуплоскость
    (c) Нижнюю полуплоскость
    (d) Единичную окружность
    (e) Единичный круг
    (f) Единичный квадрат со сторонами, параллельными осям координат

V Записать на RSL следующие определения типов с учетом ограничений типа

1. Расписание на сегодня. Содержит указание, в какой аудитории на каждой паре учится группа. Учесть, что расписание должно быть корректным, т.е. не должно быть аудитории, в которой бы занимались две разные группы.
2. Граф. Опишите константу этого типа.
3. На основе типа «Граф» описать следующие его подтипы:
    (a) ориентированный
    (b) неориентированный
    (c) полный
    (d) линейный (т.е. связный, представляющий собой цепочку вершин)
    (e) двудольный
    (f) связный
    (g) * плоский
4. Массив (представить в виде множества пар «Индекс-Значение»). Опишите константу этого типа.

5. Конечный автомат
   Примечание: Конечный автомат состоит из

   (a) множества входных символов

   (b) множества выходных символов

   (c) множества состояний

   (d) начального состояния

   (e) таблицы переходов автомата (старое состояние $\times$ символ $\rightarrow$ новое состояние $\times$ символ)

6. Машина Тьюринга
   Примечание: Машина Тьюринга состоит из

   (a) множества символов

   (b) множества состояний

   (c) пустого символа

   (d) начального состояния

   (e) множества финальных состояний

   (f) лента

   (g) таблицы автомата (старое состояние $\times$ символ $\rightarrow$ новое состояние $\times$ символ $\times$ сдвиг головки автомата)

   Запишите в типе ограничение о том, что перед началом работы слово на ленте должно иметь конечную длину (конечное число символов, отличных от пустого).

7. Измеримое множество на прямой
   Примечание: Множество E измеримо, если $\forall \varepsilon > 0 \; \exists G \supseteq E : |G \setminus E|^\star < \varepsilon$. Внешняя мера $|E|^\star = \inf \sigma(S(E))$. $\sigma(\bigcup [a_i, b_i]) = \sum |b_i - a_i|$. $S(E)$ – множество интервалов, покрывающих E.

VI Построить явную (неявную) спецификации на RSL для следующих программ

В этих задачах, если удаётся, следует привести два варианта решения: с использованием сокращённой записи множества и с помощью рекурсии.

1. Дано множество чисел. Вернуть множество квадратов чисел данного множества.

2. Дано множество чисел. Вернуть множество чисел, представимых суммами каких-либо двух элементов исходного множества.

3. Дано множество точек, заданных координатами в плоской декартовой системе координат. Получить проекцию этого множества на ось Ox.

4. Дано множество простых чисел (см. задачу IV.5). Для данного числа вернуть множество его простых делителей.

5. Дано множество чисел. Дано число. Существует ли подмножество данного множества чисел, сумма элементов которого, равна данному числу?

6. Дано множество множеств. Вернуть множество всех элементов внутренних множеств.

7. *(Задача о рюкзаке)* Дано множество предметов. Каждый предмет имеет массу. Есть n рюкзаков. Каждый рюкзак имеет вместимость K кг. Распределить все предметы по рюкзакам, не превышая вместимости каждого рюкзака.

8. Рабочая группа компьютеров задаётся IP-адресом (4 целых числа, каждое от 0 до 255) и маской (целое число от 0 до 32). Программа по данному заданию строит множество всех возможных в ней IP-адресов.

9. Для типа «Массив», определённого в задаче V.4, написать спецификации следующих операций (все изменяемые параметры должны быть включены в возвращаемое значение):

   (a) Вычисление длины массива

   (b) Получение значения по индексу i (если по этому индексу значения нет, возвратить **chaos**)

   (c) Изменение значения по индексу i на значение v (обратите внимание, что операция частично вычислима!)

   (d) Для массива из чисел: получение номера максимального элемента

10. * Для типа «Машина Тьюринга», определённого в задаче V.6, написать спецификацию следующих функций:

(a) *(загрузка слова)* по данному слову, представляющему список символов, возвращает ленту, на которой это слово расположено

(b) *(шаг машины)* по данному состоянию машины возвратить следующее состояние

(c) *(терминирование)* проверка, является ли состояние машины финальным

(d) *(применимость)* проверка, применимо ли данное слово (список символов) на данной Машине Тьюринга (если не применимо, возвратить **chaos**)

VII База данных «Журналы-Подписчики»

Для приведённого ниже неформального описания поведения функций и ограничений для базы данных «Журналы-Подписчики» составить сигнатуры и явные спецификации на RSL

```
type Magazin, Subscriber,
    Mailbox = Subscriber >< Magazin-set,
    PubHouse = Magazin-set >< Mailbox-set
value
    /* 1. выдать все журналы, печатаемые данным издательством */

    /* 2. печатает ли данное издательство данный журнал */

    /* 3. любой подписчик данного издательства выписывает
        не менее 1 и не более 5 журналов в данном издательстве */

    /* 4. любой журнал данного издательства выписывается не
            менее 1м и не более 100 людьми */

    /* 5. выдать подписчиков данного журнала
            в данном издательстве */

    /* 6. выдать множество журналов, выписываемое данным
        человеком в данном издательстве */

    /* 7. добавить журнал в данное издательство */

    /* 8. убрать журнал из данного издательства */
```

VIII Написать спецификацию согласно приведённому требованию

1. Библиотека оперирует с книгами и читателями. Написать спецификацию базы данных «Картотека», состоящая из карточек. На каждой карточке написано имя читателя и книги, которые он в данный момент читает. При описании учтите, что для одного читателя не может быть двух карточек с его именем.

2. Новый оперный театр заказал Вам спецификацию базы данных своего репертуара и труппы. Заказчик так описал Вам свой театр: есть артисты, есть оперы. Постановка задаётся указанием оперы и задействованных в ней артистов. Театр можно описать, указав труппу (артистов) и репертуар (все постановки). К базе должны быть определены следующие операции:

   (a) выдать всех артистов, работающих в театре

   (b) выдать всех артистов, участвующих в постановках

   (c) выдать все постановки, в которых задействованы не менее 20 артистов

   (d) выдать все постановки, в которых участвует данный артист

   (e) выдать всех артистов, задействованных в данной постановке

   (f) добавить нового артиста в театр

   (g) удалить артиста из театра

IX Предложить уточнение поведения

Дана аксиоматическая спецификация. Предложить уточнение для неё в виде неявной или явной спецификаций.

1.
```
type S, E
value
    f : S >< E -> Bool,
    g : S >< S -> Bool,
    h : S >< E -~-> Nat
axiom
    [A]
    all s1, s2: S :-
    (
        (all e: E :- ( f(s1, e) => f(s2, e) )
```

```
        => g(s1, s2))
    ),

    [B]
    all s: S, e: E :- ( h(s, e) is chaos ) is ~f(s, e),

    [C]
    all s: S :- ~(exists e1, e2: E :- f(s, e1) /\ f(s, e2)
        /\ h(s, e1) = h(s, e2) /\ e1 ~= e2),

    [D]
    all s: S :- ~(exists s2: S :-
        (exists e: E :- f(s, e) /\ f(s2, e)
            /\ h(s, e) ~= h(s2, e))
        => s ~= s2)
```

2. та же спецификация, но без аксиомы C

3. та же спецификация, но без аксиомы D

4. ```
   type S, E
   value
       f: S >< E -> Bool,
       u: S >< S -> S,
       i: S >< S -> S,
   axiom
       all s1, s2: S, e: E :-
           f(u(s1, s2), e) is f(s1, e) \/ f(s2, e),
       all s1, s2: S, e: E :-
           f(i(s1, s2), e) is f(s1, e) /\ f(s2, e)
   ```

5. ```
   type S, E
   value
       C: S,
       c: S -~-> Int,
       f: S >< E -> Bool,
       g: S >< E -~-> S
   axiom
       all e: E :- ~f(C, e),
       all s: S, e: E :-
   ```

```
    f(s, e) =>
    (
        let s1 = g(s, e) in
        (
            ~f(s1, e) /\
            (all e1: E :- e1 ~= e /\ f(s, e1) => f(s1, e1))
        )
        end
    ),
    c(C) is 0,
    all s: S, e: E :- c(s) is c(g(s, e)) + 1
```

6. ```
   type S, E
   value
       f: S >< E -> Bool,
       u: S >< S -> S,
       i: S >< S -> S
   axiom
       all s1, s2: S, e: E :-
           f(u(s1, s2), e) is f(s1, e) /\ ~f(s2, e),
       all s1, s2: S, e: E :-
           f(i(s1, s2), e) is f(s1, e) ~= f(s2, e)
   ```

X Разное

1. Студент Иванов написал такое определение на RSL того факта, что [0, 1] – измеримое на прямой множество (extmes : ((**Real**>< **Real**)**-infset**→ **Real**)):

   ```
   all `epsilon:Real :- `epsilon > 0 =>
     exists G: (Real >< Real)-infset :- G > E =>
       extmes( G / E ) < epsilon
   ```

   Помогите Иванову получить зачёт по МФСП, т.е. сделать так, чтобы RSL TYPE CHECKER на его спецификации не выдавал сообщений об ошибках.
   ```

Глава 4

Списки

4.1 Определения и операции

Список – это контейнер элементов одного типа, который обладает свойствами упорядоченности элементов. Для задания списка надо указать не только сами элементы, но и их порядок.

Типовое выражение для конечного списка: $typeexpr^\star$. Типовое выражение для бесконечного списка: $typeexpr^\omega$. Конечный список – подтип бесконечного списка.

Операции:

$=: T^\omega \times T^\omega \to$ **Bool** – проверка на равенство

$\neq: T^\omega \times T^\omega \to$ **Bool** – проверка на не равенство

$(.) : T^\omega \times$ **Int**$\widetilde{\to} T$ – взятие элемента по индексу (**chaos** для индекса, отсутствующего в списке, индексы нумеруются **с единицы**)

$^\wedge : T^\star \times T^\omega \to T^\omega$ – конкатенация списков

hd : $T^\omega \widetilde{\to} T$ – головной элемент списка (**chaos** для пустого списка)

tl : $T^\omega \widetilde{\to} T^\omega$ – хвостовая часть списка (**chaos** для пустого списка)

len : $T^\omega \widetilde{\to}$ **Nat** – количество элементов списка (**chaos** для бесконечного списка)

elems : $T^\omega \to T$**-infset** – множество элементов списка (без повторений!)

inds : $T^\omega \to$ **Nat-infset** – множество индексов элементов списка

Конструктор списка:

(*пустой список*) $\langle\rangle$

(*перечисление*) $\langle 0, 1, 2\rangle$ - список, состоящий из трех целых чисел (не натуральных!) - нуля, единицы и двойки - в порядке увеличения.

(*диапазон*) $\langle 0..2\rangle$ - то же, что и $\langle 0, 1, 2\rangle$; $\langle 0..0\rangle \equiv \langle 0\rangle$, $\langle 1..0\rangle \equiv \langle\rangle$

(*«сокращенная запись»*) $\langle expr\_with\_var | var$ **in** $listexpr \bullet boolexpr\rangle$ (например, $\langle 2 \star n | n$ **in** $\langle 0..2\rangle\rangle$, что эквивалентно $\langle 0, 2, 4\rangle$)

4.2 Задачи

I Вычислить

1. $\langle 1\rangle = \langle 1, 1\rangle$
2. $\langle 1, 2\rangle = \langle 3, 1\rangle$
3. $\langle 1, 2\rangle = \langle 2, 1\rangle$
4. $\langle 1, 2, 3\rangle(2)$
5. $\langle 1\rangle(0)$
6. $\langle 1\rangle(1)$
7. $\langle 1, 2\rangle \,\hat{}\, \langle 3, 4\rangle$
8. $\langle 1, 2\rangle \,\hat{}\, \langle 2, 3\rangle$
9. $\langle x + 10 | x$ **in** $\langle 1, 2\rangle\rangle = \langle x | x$ **in** $\langle 1\rangle\rangle$
10. **hd**$\langle 1, 2, 3\rangle$
11. **tl**$\langle 1, 2, 3\rangle$
12. **len**$\langle 1, 2, 3\rangle$
13. **elems**$\langle 1, 2, 3\rangle$
14. **inds**$\langle 1, 2, 3\rangle$
15. **len**$\langle 1..30\rangle$

16. **let** $x = \langle 1, 2, 3 \rangle$ **in** **elems** $x \cap$ **inds** x **end**
17. **let** $x = \langle 0, 1, 2 \rangle$ **in** **elems** $x \cap$ **inds** x **end**
18. **let** x : **Int**$^\omega$ $\bullet (\forall\, i_1, i_2$: **Nat** $\bullet$ **card**$\{i_1, i_2\} =$ **card**$\{x(i_1), x(i_2)\})$ **in** x **end**
19. **let** x : **Int**$^\omega$ $\bullet (\forall\, i_1, i_2$: **Nat** $\bullet$ **card**$\{i_1, i_2\} =$ **card**$\{x(i_1), x(i_2)\})$ **in** **elems** $x \cap$ **inds** x **end**
20. $\forall\, x : T^\omega \bullet x(0) =$ **hd** x
21. $\forall\, x : T^\omega \bullet x(0) \equiv$ **hd** x

II Решите уравнения

1. $\langle 1 \rangle \,^\frown x = \langle 1 \rangle$
2. $\langle 1 \rangle \,^\frown x = \langle 1, 2, 3 \rangle$
3. $\langle 1 \rangle \,^\frown x = \langle 3, 2, 1 \rangle$
4. **tl** $x = \langle 1 \rangle$
5. **hd** $x = 1$
6. $x \,^\frown$ **tl** $x = x$
7. **tl** $x = \langle$ **hd** $x \rangle$
8. **elems** $x =$ **inds** x
9. **elems tl** $x =$ **elems** x
10. **len** $x =$ **card elems** x
11. **len** $x =$ **card inds** x
12. **len** $x \equiv$ **card inds** x

III Записать на RSL следующие константы и определения типов

1. Пустой список
2. Список из чисел 1, 2, 3; сколько решений имеет эта задача?
3. Список всех простых натуральных чисел в порядке увеличения их значения
4. Список всех пар взаимнопростых натуральных чисел в порядке увеличения их суммы
5. Список номеров групп 4го курса факультета ВМиК МГУ в порядке увеличения номера группы
6. Количество простых чисел от 1 до 10 (тремя различными способами)

IV Напишите явные (неявные) спецификации следующих функций

1. Определить, является ли бесконечным данный список.
2. Вычислить длину списка без использования функции **len**.
3. Вычислить сумму элементов списка.
4. Вычислить произведение элементов списка.
5. Дан список. Построить список из элементов исходного списка, элементы которого идут в обратном порядке по отношению к исходному списку
6. Дана строка и символ. Определить, встречается ли символ в данной строке (предложить два различных способа решения).
7. Дана строка. Определить самый часто встречающийся в ней символ.
8. Отсортировать данный список вещественных чисел в порядке возрастания.
9. Построить список всех целых чисел в порядке неубывания модуля.

10. Определить sup списка вещественных чисел (в случае конечного списка это будет и максимум).

11. Выдать число, не встречающееся в данном списке.

12. Дан список чисел. Построить по нему список квадратов, расположив элементы

    (a) с сохранением порядка исходного списка

    (b) в порядке убывания модуля

    (c) так, чтобы не было трёх подряд чисел, расположенных в порядке возрастания или убывания.

13. Дано натуральное число. Построить список степеней его простых делителей, в котором на местоположение степени есть её основание (а значение, соответственно, показатель)

14. Дан список показателей степеней (местоположение – основание степени). Построить соответствующее натуральное число.

15. Дан список натуральных чисел, отличных от нуля. Вернуть список, в котором на месте №i находится количество раз, которое i встречается в исходном списке.

16. Дано множество чисел. Построить из него список, расположив элементы множества

    (a) в порядке убывания

    (b) в порядке убывания модуля

17. Дан список чисел. Построить из него новый список, расположив элементы исходного списка

    (a) в порядке убывания

    (b) в порядке убывания модуля

18. Дан список из чисел. Построить из него множество, использовав все элементы данного списка.

19. Дан множество пар (ключ, объект) и список ключей. Построить соответствующий ему список объектов. Считайте, что во множество пар ключи не повторяются.

20. Дан список чисел. Можно ли суммой некоторых его элементов получить

    (a) четное число (для списка из целых чисел)

    (b) простое число (для списка из целых чисел)

    (c) целое число (для списка из вещественных чисел)

V Упростить выражения

1. $\langle x \mid x$ **in** $\langle$**card**$\{a..b\}\rangle\rangle$
2. $\langle c \mid c$ **in** $\langle'a', 'b'\rangle\rangle$
3. $\langle c \mid c$ **in** $\langle' "'\rangle\rangle$
4. $\langle c \mid c$ **in** $\langle\rangle\rangle$

VI Опишите на RSL необходимые типы и спецификации функций для следующих систем

1. Опишите модуль ARRAY, обеспечивающий работу с массивами чисел длины N. Модуль должен поддерживать следующие функции:

    (a) выдать длину массива

    (b) выдать значение массива с по данному индексу

    (c) сложить два массива

    (d) скалярно перемножить два массива

2. Опишите модуль STRING со следующими функциями

    (a) проверка строки на пустоту

    (b) константа – пустая строка

    (c) длина строки

    (d) конкатенация строк

    (e) проверка вхождения подстроки в данную строку

    (f) выдача позиции первого вхождени подстроки в данную строку

    (g) выдача позиции последнего вхождения подстроки в данную строку

(h) выдача количества вхождений данной подстроки в данную строку

(i) замена всех вхождений данной подстроки в данную строку, начиная с самого первого без наложений, на другую данную строку

(j) убрать из строки начальные и завершающие её пробелы

(k) составить список слов, входящих в данную строку. Слова разделяются пробелом, запятой или точкой.

3. Опишите модуль «числовая последовательность» со следующими операциями, возвращающими **Bool**

   (a) последовательность ограничена

   (b) последовательность монотонно возрастает

   (c) последовательность сходится

   (d) * числовой ряд из элементов последовательности (взятых в том же порядке, что и в последовательности) сходится

4. Опишите модуль PAGE, обеспечивающий работу со страницей текста. Текст есть список строк. Строка есть список слов. Модуль должен поддерживать следующие функции:

   (a) проверка того, что встречается ли данное слово на странице

   (b) количество вхождений данного слова на странице

5. Опишите модуль поддержки типа-контейнера, в котором все элементы различны и не упорядочены со следующими функциями

   (a) проверка контейнера на пустоту

   (b) константа – пустой контейнер

   (c) количество элементов в контейнере

   (d) добавление в контейнер элемента

   (e) проверка вхождения элемента в контейнер

   (f) проверка того, что все элементы первого данного контейнера присутствуют во втором данном контейнере

   (g) построить контейнер, содержащий общие элементы данных двух контейнеров

   (h) построить контейнер, содержащий все элементы, встречающиеся хотя бы в одном из двух данных контейнеров

6. Опишите модуль поддержки типа-контейнера, в котором операция удаления изымает последний добавленный элемент, со следующими функциями

   (a) проверка контейнера на пустоту

   (b) константа – пустой контейнер

   (c) количество элементов в контейнере

   (d) добавление в контейнер элемента

   (e) удаление элемента из контейнера согласно приведённой семантике

   (f) проверка вхождения элемента в контейнер

7. Опишите модуль поддержки типа-контейнера, в котором операция удаления изымает первый добавленный элемент, со следующими функциями

   (a) проверка контейнера на пустоту

   (b) константа – пустой контейнер

   (c) количество элементов в контейнере

   (d) добавление в контейнер элемента

   (e) удаление элемента из контейнера согласно приведённой семантике

   (f) проверка вхождения элемента в контейнер

8. На основе множества реализуйте на RSL контейнер-список со следующими функциями

   (a) проверка контейнера на пустоту

   (b) константа – пустой контейнер

   (c) длина контейнера

   (d) конкатенация двух данных контейнеров

   (e) получение головы контейнера

   (f) получение хвоста контейнера

   (g) получение множества элементов контейнера

9. Матрица задаётся списком списков одинаковой длины. Определите тип «Матрица» и его подтипы:

(a) Квадратная матрица

(b) Симметричная матрица

(c) Верхнетреугольная матрица

(d) * Невырожденная матрица
Примечание: Следующие аксиомы[1] определяют единственную операцию – детерминант матрицы:

i. если одна из строк матрицы умножается на число k, то детерминант умножается на k

ii. детерминант не меняется, если к одной из строк прибавляется другая строка

iii. детерминант единичной матрицы равен 1

Для типа «Матрица» определите следующие операции

(a) сложение двух данных матриц

(b) вычитание двух данных матриц

(c) произведение двух данных матриц

(d) транспонирование данной матрицы

10. Граф задаётся своей матрицей смежности. На основе задачи 9 определите тип «Граф» и следующие его подтипы:

(a) ориентированный

(b) неориентированный

(c) полный

(d) линейный (т.е. связный, представляющий собой цепочку вершин)

(e) двудольный

(f) связный

(g) * плоский

11. Определите тип «Машина Тьюринга»
Примечание: Машина Тьюринга состоит из

(a) множества символов

(b) множества состояний

[1] см. Курош. Курс высшей алгебры

(c) пустого символа

(d) начального состояния

(e) множества финальных состояний

(f) лента (бесконечный список)

(g) таблицы автомата (старое состояние $\times$ символ $\rightarrow$ новое состояние $\times$ символ $\times$ сдвиг головки автомата)

Запишите в типе ограничение о том, что перед началом работы слово на ленте должно иметь конечную длину (конечное число символов, отличных от пустого).

12. «Система учёта продажи билетов в кинотеатр». Зал состоит из N рядов по M мест в каждом. Сеанс задаётся днём (количество дней, прошедшее с «сегодня» - при наступлении нового дня все номера дней уменьшаются на единицу), названием и информацией о проданных билетах ($= 0 \Rightarrow$ билет на это место продан; $> 0 \Rightarrow$ не продано, значение и есть цена билета на это место). Написать определение всех необходимых типов и следующих операций:

    (a) выдать информацию о непроданных билетах на сегодняшний сеанс

    (b) выдать ближайший показ фильма с данным названием

    (c) выдать название фильма, идущего сегодня

    (d) есть ли места на ближайший показ фильма с данным названием

13. Реализуйте модуль ассоциативного поиска со следующими операциями

    (a) очистка содержимого поиска

    (b) добавление пары (ключ, значение) (с проверкой уникальности добавляемого ключа)

    (c) проверка вхождения данного ключа

    (d) поиск значения по заданному ключу

14. Реализуйте модуль хэшированного хранения данных (размер хэш-таблицы равен N, хэш-функция $f(x) = x \bmod M$, M – простое число; при совпадении значения хэш-функции элемент помещается в конец списка для данного значения хэш-функции) со следующими операциями

(a) очистка содержимого хранения

(b) добавление пары (ключ, значение)

(c) проверка вхождения данного ключа

(d) поиск значения по заданному ключу

15. * Реализуйте простой «менеджер динамической памяти». Память представить в виде списка значений. Введите список свободной памяти и список занятой памяти. Менеджер выдаёт только непрерывные куски памяти. Если подходящего куска нет, то менеджер выдаёт **chaos**. Менеджер должен поддерживать следующие операции:

    (a) new : Amount $\xrightarrow{\sim}$ Pointer - выделить новый объем памяти

    (b) delete : Pointer $\xrightarrow{\sim}$ **Unit**- отдать кусок памяти, начало которого задаётся аргументом; возвращает **chaos**, если память по этому адресу уже была освобождена

VII Уточнение алгебраической спецификации

Для данных аксиоматических спецификаций постройте явные или неявные спецификации

1.
```
type
    T, L
value
    C:L,
    f: L >< L -> L,
    g: L -~-> T
axiom
    f(C, C) is C,
    g(C) is chaos,
    all x: L :- f(C, x) is x,
    all x: L :- f(x, C) is x,
    all x, y, z: L :- f(x, f(y, z)) is f(f(x, y), z),
    all x, y : L :- x ~= C => (g(f(x, y)) is g(x))
```

2.
```
type
    T, L
value
    C: L,
```

```
    f: L -> Nat,
    g: L >< T -> L
axiom
    f(C) is 0,
    all x: L, y: T :- f(g(x, y)) = f(x) + 1,
    all x: L, y: T :- g(x, y) ~= C,
    all x1, x2: L, y1, y2: T :- g(x1, y1) = g(x2, y2)
        => (x1 = x2 /\ y1 = y2)
```

3.
```
type
    T, L
value
    f: L >< T -> L,
    g: L -~-> L
axiom
    all x: L, y: T :- g(f(x, y)) is x,
    all x1, x2: L, y1, y2: T :- f(x1, y1) = f(x2, y2)
        => (x1 = x2 /\ y1 = y2)
```

4.
```
type
    A, C
value
    empty: C,
    add: A >< C -> C,
    f: C -> C,
    g: C -> Bool
axiom
    f(empty) is empty,
    all x: C, y, z: A :-
        f(add(add(x, y), z)) is add(add(f(x), z), y)
        pre g(z),
    g(empty) is true,
    all x: C, y, z: A :- g(add(add(x, y), z)) is g(x)
```

Глава 5

Отображения

5.1 Определения и операции

Отображение – это множество пар элементов, у которого первые компоненты не повторяются.

Типовое выражение для детерминированного отображения: $typeexpr_1 \underset{m}{\longrightarrow} typeexpr_2$.

Типовое выражение для недетерминированного отображения: $typeexpr_1 \underset{m}{\widetilde{\longrightarrow}} typeexpr_2$.

Операции:

$=: (T_1 \underset{m}{\longrightarrow} T_2) \times (T_1 \underset{m}{\longrightarrow} T_2) \to$ **Bool** – сравнение отображений на равенство

$\neq: (T_1 \underset{m}{\longrightarrow} T_2) \times (T_1 \underset{m}{\longrightarrow} T_2) \to$ **Bool** – сравнение отображений на не равенство

$(.) : (T_1 \underset{m}{\longrightarrow} T_2) \times T_1 \widetilde{\to} T_2$ – взятие значения по индексу (**chaos**, если значение по этому индексу не определено)

dom $: (T_1 \underset{m}{\longrightarrow} T_2) \to T_1$**-infset** – *область определения* отображения

rng $: (T_1 \underset{m}{\longrightarrow} T_2) \to T_2$**-infset** – *область значения* отображения

$\dagger : (T_1 \underset{m}{\longrightarrow} T_2) \times (T_1 \underset{m}{\longrightarrow} T_2) \to (T_1 \underset{m}{\longrightarrow} T_2)$ – обновление отображения

$\cup : (T_1 \underset{m}{\longrightarrow} T_2) \times (T_1 \underset{m}{\longrightarrow} T_2) \to (T_1 \underset{m}{\widetilde{\longrightarrow}} T_2)$ – объединение отображений

$\setminus : (T_1 \underset{m}{\longrightarrow} T_2) \times T_1$**-infset** $\to (T_1 \underset{m}{\longrightarrow} T_2)$ – уменьшение отображения

$/ : (T_1 \underset{m}{\longrightarrow} T_2) \times T_1$**-infset** $\to (T_1 \underset{m}{\longrightarrow} T_2)$ – проекция отображения

$\circ : (T_2 \underset{m}{\longrightarrow} T_3) \times (T_1 \underset{m}{\longrightarrow} T_2) \to (T_1 \underset{m}{\longrightarrow} T_2)$ – композиция отображений

Конструктор отображения:

(*пустое отображение*) []

(*перечисление*) $[0 \mapsto 1, 1 \mapsto 2]$ – отображение, состоящий из двух пар целых чисел (не натуральных!) - из 0 в 1 и из 1 в 2.

(*«сокращенная запись»*) $[expr_1\_with\_var \mapsto expr_2\_with\_var | var : typeexpr \bullet boolexpr]$ (например, $[n \mapsto n+1 | n : \mathbf{Nat} \bullet n < 3]$, что эквивалентно $[0 \mapsto 1, 1 \mapsto 2, 2 \mapsto 3]$)

5.2 Задачи

I Вычислить

1. $[n \mapsto 1 | n : \mathbf{Nat} \bullet n \in \{1..3\}]$
2. $[n \mapsto n | n : \mathbf{Nat} \bullet n \in \{5..5\}]$
3. $[n \mapsto n+1 | n : \mathbf{Nat} \bullet n \in \{100..90\}]$
4. $[n \mapsto m | n, m : \mathbf{Nat} \bullet n \setminus m = 0 \wedge m > 2]$
5. $[n \mapsto m | n, m : \mathbf{Nat} \bullet n \in \{1..3\} \wedge m \in \{1..n\}]$
6. $[n \mapsto (p, q) | n, p, q : \mathbf{Nat} \bullet n \in \{1..100\} \wedge p + q = n \wedge p \setminus q = 0]$
7. $[1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 1](3)$
8. $[1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 1](4)$
9. ["Маша" $\mapsto 30$, "Света" $\mapsto 15$, "Маша" $\mapsto 30$]("Маша")
10. ["Маша" $\mapsto 30$, "Света" $\mapsto 15$, "Маша" $\mapsto 10$]("Маша")
11. $[1 \mapsto [1 \mapsto 1,\ 2 \mapsto 2,\ 3 \mapsto 3], 2 \mapsto [1 \mapsto 4,\ 2 \mapsto 5,\ 3 \mapsto 6]\ ]\ (2)\ (1)$

12. $\mathbf{dom}[3 \mapsto 1, 5 \mapsto 0, 2 \mapsto 88]$
13. $\mathbf{rng}[3 \mapsto 1, 5 \mapsto 0, 2 \mapsto 88]$
14. $\mathbf{dom}[n \mapsto 2 * n | n : \mathbf{Nat}]$
15. $\mathbf{rng}[n \mapsto 2 * n | n : \mathbf{Nat}]$
16. $[1 \mapsto 20, 2 \mapsto 30] \cup [1 \mapsto 30, 2 \mapsto 20]$
17. $[1 \mapsto 20, 2 \mapsto 30] \dagger [1 \mapsto 30, 2 \mapsto 20]$
18. $["\text{Миша}" \mapsto 170, "\text{Слава}" \mapsto 200, "\text{Витя}" \mapsto 195] \setminus \{"\text{Миша}", "\text{Петя}"\}$
19. $["\text{Миша}" \mapsto 170, "\text{Слава}" \mapsto 200, "\text{Витя}" \mapsto 195] \,/\{"\text{Миша}", "\text{Петя}"\}$
20. Пусть Friend = $["\text{Миша}" \mapsto "\text{Аня}", "\text{Аня}" \mapsto "\text{Леша}", "\text{Леша}" \mapsto "\text{Миша}"]$. Найти Friend $\circ$ Friend. Какой смысл этого значения ?
21. $\mathbf{card}\,\mathbf{dom}[1 \mapsto 2, 1 \mapsto 3, 2 \mapsto 3]$
22. $\mathbf{dom}([10 \mapsto 100, 20 \mapsto 50] \dagger [20 \mapsto 60, 30 \mapsto 90])$
23. $\mathbf{card}\,\mathbf{rng}([1 \mapsto 2] \circ [3 \mapsto 2, 4 \mapsto 1] \circ [1 \mapsto 2, 3 \mapsto 4])$

II Какие из следующих выражений верны

Если выражение неверно, привести контпример и дополнительные ограничения на входящие переменные, чтобы условие стало верным. Считать, что все переменные стоят под кванторами всеобщности.

1. $\mathbf{dom}(X \cup Y) \equiv \mathbf{dom}\,X \cup \mathbf{dom}\,Y$
2. $\mathbf{rng}(X \cup Y) \equiv \mathbf{rng}\,X \cup \mathbf{rng}\,Y$
3. $\mathbf{dom}(X \dagger Y) \equiv \mathbf{dom}\,X \dagger \mathbf{dom}\,Y$
4. $\mathbf{rng}(X \dagger Y) \equiv \mathbf{rng}\,X \dagger \mathbf{rng}\,Y$
5. $X = Y \Rightarrow \mathbf{dom}\,X = \mathbf{dom}\,Y$
6. $X \neq Y \Rightarrow \mathbf{dom}\,X \neq \mathbf{dom}\,Y$
7. $X \neq Y \Rightarrow \mathbf{rng}\,X \neq \mathbf{rng}\,Y$
8. $\mathbf{dom}(X \setminus Y) \equiv (\mathbf{dom}\,X\ ) \setminus Y$

9. $\mathbf{dom}(X/Y) \equiv Y$
10. $\mathbf{dom}(X \dagger Y) \equiv \mathbf{dom}\, X \cup Y$
11. $(X \cup Y) \cup Z \equiv X \cup (Y \cup Z)$
12. $(X \dagger Y) \dagger Z \equiv X \dagger (Y \dagger Z)$
13. $(X \circ Y) \circ Z \equiv X \circ (Y \circ Z)$
14. $X \cup Y \equiv Y \cup X$

III Записать на RSL следующие константы и определения типов

1. Записать отображение – перестановку первых N натуральных чисел. Считать N константой с sort-определением.
2. Записать отображение-«сдвиг» : $[1 \mapsto 2, 2 \mapsto 3, ..., N \mapsto 1]$. Считать N константой с sort-определением.
3. Записать отображение всех полных квадратов в своё основание. $[1 \mapsto 1, 4 \mapsto 2, 9 \mapsto 3, ...]$
4. Записать отображение из любого натурального числа в его простой делитель.
5. Записать отображение любого натурального числа в большее его простое натуральное число. $[1 \mapsto 5, 2 \mapsto 7, 3 \mapsto 5, ...]$
6. Записать отображение «бесконечная перестановка».
7. Записать отображение любого натурального числа в своё «зеркало» – число из цифр исходного числа, записанных в обратном порядке. $[1 \mapsto 1, ..., 12 \mapsto 21, ..., 832 \mapsto 238, ...]$
8. Записать отображение любого текста в его реверсию. [..., "стол" $\mapsto$ "лотс", "книга" $\mapsto$ "агинк", ...]
9. Не меняя описания предыдущей константы, описать новое отображение любого текста, начинающегося с 'а', в его реверсию.
10. Записать тип «Англо-русский словарь» так, как Вы его представляете. Учтите, что слово может иметь несколько переводов.

IV Запишите спецификацию функций в явном (неявном) виде

1. Подсчитать количество элементов в данном отображении, если оно
   (a) детерминированное
   (b) * недетерминированное
2. По отображению $\{[a \mapsto b]\}$ построить отображение $\{[a^2 \mapsto b^2]\}$.
3. Дано отображение **Nat** $\underset{m}{\longrightarrow}$ **Nat**. Вернуть количество элементов, отображающих в 0.
4. По данному отображению $\{[a \mapsto b]\}$ построить отображение $\{[b \mapsto x]\}$, где b - правая часть некоторого элемента исходного отображения, а x - количество раз, которое b встретилось в исходном отображении.
5. Дано отображение **Nat** $\underset{m}{\longrightarrow}$ **Nat**. Вернуть количество различных элементов, в которые осуществляется отображение.
6. Дано отображение **Nat** $\underset{m}{\longrightarrow}$ **Real**, представляющее основание и показатель степени в разложении числа на простые множители. Вернуть число, которое представлено таким отображением.
7. Дано натуральное число. Построить по нему отображение **Nat** $\underset{m}{\longrightarrow}$ **Real**, представляющее основание и показатель степени в разложении его на простые сомножители.
8. Дано отображение строк $\{[s \mapsto t]\}$. Оставить в нём только те элементы, левая часть которого начинается с s_0.
9. Проверить, является ли данное отображение детерминированным.
10. Проверить, является ли данное отображение конечным.
11. Проверить, является ли данное отображение взаимнооднозначным.
12. Проверить, есть ли в данном отображении элемент $x \mapsto y$, где x – наибольший из всех левых частей, а y – наименьший среди всех правых частей.
13. Проверить, является ли данное отображение перестановкой.

14. Проверить, является ли данное отображение перестановкой первых N натуральных чисел.

15. Дано отображение $\{[(x, y) \mapsto L]\}$ (L – множество). Проверить, верно ли, что все элементы в L не больше y и не меньше x.

16. Дано отображение. Построить его максимальное детерминированное подотображение.

17. Реляционное отношение задано отображением $T \underset{m}{\longrightarrow} A \times B \times C$. Проверить, есть ли среди атрибутов A, B, C возможные ключи. Домены атрибутов считать sort-определенными.

18. Реализуйте функцию $lower$, переводящую символы в нижний регистр. Какова, по Вашему, будет сигнатура этой функции?

V Дайте спецификацию на RSL для следующих систем

1. Склад товаров можно представить в виде отображения: Имя_товара $\mapsto$ Количество_товара. Опишите тип «Склад товаров» со следующими операциями (представленные операции получают на вход склад и возвращают новый склад):
   (a) выдать все имеющиеся на складе товары
   (b) взять товар (его некоторое количество) со склада
   (c) добавить товар (его некоторое количество) на склад
   (d) выдать все закончившиеся товары
   (e) выдать оставшееся количество товара с данным именем

2. Склад товаров можно представить в виде отображения: Имя_товара $\mapsto$ Цена_товара. Опишите тип «Склад товаров» со следующими операциями (представленные операции получают на вход склад и возвращают новый склад):
   (a) выдать все имеющиеся на складе товары
   (b) установит на товар данную цену
   (c) сбросить цены на все товары на данное количество процентов
   (d) вычислить среднюю цену на товар
   (e) выдать все дешёвые товары (цена которых меньше средней цены по складу)

3. Опишите модуль ШКОЛА: у каждого ученика данной школы есть дневник, двух одинаковых учеников нет, дневник содержит оценки по каждому предмету, предметы не повторяются. Описать в это модуле следующий набор операций:

   (a) выдать всех учеников данной школы

   (b) выдать все предметы данной школы

   (c) выдать всех отличников данной школы (тех, кто учится на одни пятёрки)

   (d) выдать процент учеников школы, не усваивающих материал (т.е. имеющих меньше трёх баллов за какой-либо предмет)

   (e) выдать предмет, который усваивается хуже всех

   (f) выдать предмет, который усваивается лучше всех

4. Описать модуль МАССИВ, содержащий описание типа «Массив» из элементов одного типа. Размер массива N (константа с sort-определением). Модуль содержит следующие операции:

   (a) выдать длину данного массива

   (b) выдать значение элемента данного массива по данному индексу

   (c) вычислить максимальное значение среди элементов данного массива

   (d) вычислить сумму элементов данного массива

   (e) отсортировать элементы массива в порядке возрастания

   (f) вернуть индекс первого вхождения в массив данного элемента x

   (g) по данному массиву построить массив из тех же элементов, но в обратном порядке

   (h) слить два данных упорядоченных по возрастанию массивов (создать новый массив из всех элементов данных двух массивов, также упорядоченный, методом слияния)

5. Описать модуль РАЗРЕЖЕННЫЙМАССИВ, содержащий описание типа «Разреженный массив» из элементов одного типа. Размер массива N (константа с sort-определением). Модуль содержит следующие операции:

   (a) выдать текущее количество элементов данного массива

(b) выдать значение элемента данного массива по данному индексу

(c) запомнить значение элемента данного массива по данному индексу

(d) вычислить максимальное значение среди элементов данного массива

(e) вычислить сумму элементов данного массива

(f) отсортировать элементы массива в порядке возрастания

(g) вернуть индекс первого вхождения в массив данного элемента x

6. Описать модуль ПОЛИНОМ, содержащий описание типа «Полином» в виде отображения { [индекс $\mapsto$ коэффициент] } (индексы с нулевым коэффициентом не представлены) и спецификацию следующих операций-задач[1]:

   (a) вернуть степень данного полинома

   (b) вернуть значение максимального коэффициента данного полинома.

   (c) вернуть значение данного полинома в данной точке x.

   (d) вернуть значение производной данного полинома в данной точке x.

   (e) вернуть сумму двух данных полиномов (это будет полином; степени данных полиномов могут быть разными)

   (f) вернуть произведение двух данных полиномов (это будет полином; степени данных полиномов могут быть разными)

   (g) вернуть НОД двух данных полиномов

7. Описать модуль МАТРИЦА, содержащий описание типа «Матрица» (например, в виде отображения натуральных чисел в отображения натуральных чисел в вещественные, не забыв ограничения на такое задание) со следующими операциями:

   (a) выдать элемент с индексами $[i, j]$ (i – номер строки, j – номер столбца)

   (b) выдать множество элементов матрицы

[1] Большинство задач взято из книги А. Шеня «Программирование: теоремы и задачи»

(c) линеаризовать множество в список (сначала первая строка, затем вторая, затем третья и т.д.)

(d) сложить две матрицы

(e) перемножить две матрицы

(f) транспонировать матрицу

8. Описать модуль LISTMAP, содержащий описание типа список на основе типа отображение со следующими операциями

(a) проверка на пустоту

(b) длина

(c) конкатенация двух данных списков

(d) получение головы списка

(e) получение хвоста списка

(f) получение множества элементов списка

9. Описать модуль SETMAP, содержащий описание типа множество на основе типа отображение со следующими операциями:

(a) проверка на пустоту

(b) количество элементов

(c) объединение множеств

(d) пересечение множеств

(e) вычитание множеств

(f) проверка того, что одно множество нестрого вложено в другое

(g) проверка того, что данный элемент входит в данное множество

10. Описать модуль MAPLIST, содержащий описание типа (детерминированное) отображение на основе типа список со следующими операциями:

(a) проверка на пустоту

(b) получение области определения

(c) получение области значений

(d) обновление данного отображения вторым данным отображением

11. Описать модуль MAPSET, содержащий описание типа (детерминированное) отображение на основе типа множество со следующими операциями:

    (a) проверка на пустоту

    (b) получение области определения

    (c) получение области значений

    (d) обновление данного отображения вторым данным отображением

12. Реализуйте на основе отображений модуль хэшированного хранения данных (размер хэш-таблицы равен N, хэш-функция $f(x) = x \bmod M$, M – простое число; при совпадении значения хэш-функции элемент помещается в конец списка для данного значения хэш-функции) со следующими операциями

    (a) очистка содержимого хранения

    (b) добавление пары (ключ, значение)

    (c) проверка вхождения данного ключа

    (d) поиск значения по заданному ключу

VI База данных «Журналы-Подписчики»

Пусть база данных издательства описана так, как показано ниже. Реализуйте функции по сигнатуре и семантике, записанной в комментарии перед сигнатурой.

```
type
    Magazin,
    Subscriber,
    Mailboxes = Subscriber -m-> Magazin-set,
    PubHouse = Magazin-set >< Mailboxes
value
    /* выдать все журналы, печатаемые данным издательством */
    allMagazines : PubHouse -> Magazin-set,

    /* печатает ли данное издательство данный журнал */
    hasMagazin : PubHouse >< Magazin -> Bool,

    /* любой человек выписывает в данном издательстве
```

```
        не менее 1 и не более 5 журналов */
    success : PubHouse -> Bool,

    /* любой журнал в данном издательстве выписывается
        не менее 1м и не более 100 людьми */
    success2 : PubHouse -> Bool,

    /* выдать всех подписчиков данного журнала в данном изд-ве */
    allSubscribers : PubHouse >< Magazin -> Subscriber-set,

    /* выдать множество журналов, выписываемое данным
        человеком в данном издательстве */
    magsOf : Subscriber >< PubHouse -> Magazin-set,

    /* добавить журнал в данное издательство */
    addMagazin : PubHouse >< Magazin -> PubHouse,

    /* убрать журнал из данного издательства */
    dropMagazin : PubHouse >< Magazin -> PubHouse
```

VII ** Система поддержки метрополитена

Дополните данное описание требованием о том, что станции на линии уникальны, и опишите тела всех функций, сигнатуры и неформальная семантика которых записаны ниже:

```
type
    СхемаМетро = НазваниеЛинии -m-> СтанцияНаСхеме-list,
    СтанцияНаСхеме = НазваниеСтанции >< Переход-set,
    Станция = НазваниеСтанции >< НазваниеЛинии,
    Переход = Станция,
    НазваниеСтанции = Text
/* По первому пути идут поезда в сторону увеличения индекса
        станции в списке, описывающем линию ;
    названия линий уникальны, названия станций могут повторяться,
    названия станций на одной линии уникальны */
value
    /* выдать множество всех линий метрополитена */
    всеЛинии : СхемаМетро -> НазваниеЛинии-set,

    /* выдать множество всех станций метрополитена */
```

```
всеСтанции : СхемаМетро -> Станция-set,

/* множество станций, до которых идут поезда с первого пути */
Поезда1Путь : СхемаМетро >< Станция -> НазваниеСтанции-set,

/* выдать текстовое сообщение при закрытии дверей на данной
    станции, если поезд находится на втором пути (название
    следующей станции при открытии дверей не объявляется);
    не забудьте о детях и инвалидах */
ДвериЗакрываются : СхемаМетро >< Станция -> Text,

/* выдать самый большой переход */
СамыйБольшойПереход : СхемаМетро -> Станция-set,

/* есть ли кольцевая линия? */
ЕстьКольцо : СхемаМетро -> Bool,

/* можно ли с кольца перейти на любую линию? */
СКольцаНаЛюбую : СхемаМетро -> Bool,

/* можно ли с любой линии перейти на любую линию? */
СЛюбойНаЛюбую : СхемаМетро -> Bool,

/* можно ли доехать от любой станции до любой
    другой, совершив не более двух пересадок? */
НормальнаяСхема: СхемаМетро -> Bool,

/* найти путь в виде списка станций между двумя
    данными станциями */
Путь: СхемаМетро >< Станция >< Станция -~->
        Станция-list,

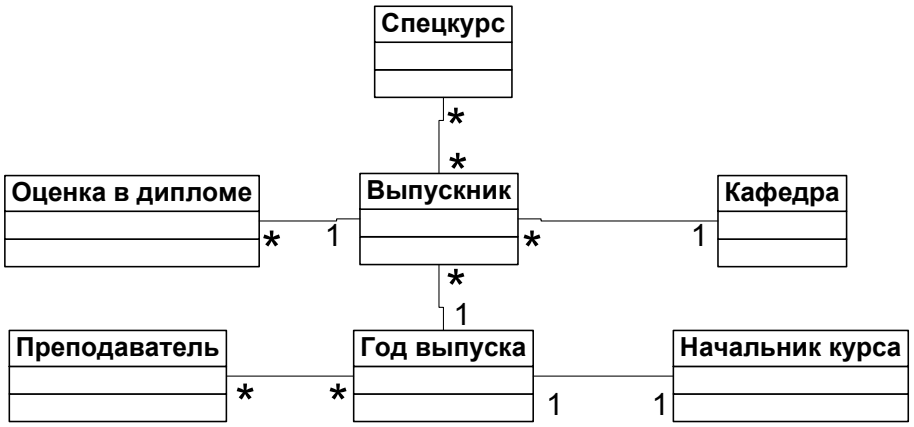
/* найти путь в виде списка станций между двумя
    данными станциями минимальной длины*/
МинимПуть: СхемаМетро >< Станция >< Станция -~->
        Станция-list,

/* между какими двумя станциями среди всех путей
    в виде списка станций минимальной длины путь
    является максимальным ? */
СамыеУдаленные: СхемаМетро -> Станция >< Станция
```

VIII ER-диаграммы

Предложите наиболее общий способ описания модели данных, представимой ER-диаграммой (1–1, 1–⋆, ⋆–⋆).

На основе Вашего предложения напишите на RSL определения типов для модели данных «Выпускники», согласно диаграмме на приведенном рисунке:



IX Запишите спецификацию функции в явном виде по её неявному заданию

1.
```
type
    X, Y, M = X -m-> Y
value
    f : M >< M -> M
    f(x, y) as z
    post
        dom z = dom x union dom y
        /\
        (
            all a : X :-
                a isin dom z =>
                z(a) = (if a isin dom y then y else x end)(a)
        )
```

2.
```
type
    X, Y, M = X -m-> Y
value
```

```
    f: M >< X-infset -> M
    f(x, y) as z
    post
        dom z = dom x inter y
        /\
        (
            all a : X :- a isin dom z => z(a) = x(a)
        )
```

3.
```
type
    X, Y, M = X -m-> Y
value
    f: M >< X-infset >< X-infset -> M
    f(x, y, z) as t
    post
        dom t = dom x \ (y union z)
        /\
        (
            all a : X :- a isin dom t => z(a) = x(a)
        )
```

Глава 6

Недетерминизм. Параллелизм.

Формальное определение операций смотрите в [2].

6.1 Модель параллельных вычислений в RSL

Основное понятие – *вычислитель*. По-другому, вычислитель можно называть *выражением*. Поэтому каждый вычислитель имеет свою *запись* – вычислитель можно записать на языке RSL. Примеры вычислителей (каждый вычислитель записан в кавычках, которые не являются частью вычислителя): «1», «1 + 3», «x := 1».

Канал – некая именованная сущность, для которой определены 2 операции: послать в канал, считать из канала. В RSL для каждой из этих операций определён свой вычислитель со следующим синтаксисом:

- «имя_канала ! вычислитель» – послать в канал с именем «имя_канала» результат вычисления «вычислител»я, результат вычисления данного вычислителя имеет тип **Unit**
- «имя_канала ?» – считать из канала с именем «имя_канала» нечто, что и будет результатом вычисления этого вычислителя

В один момент времени канал может хранить только одно значение определённого типа (и тип этот меняться во времени не может). Любая посылка в занятый канал будет блокироваться до тех пор, пока канал не будет освобождён. Любой приём из канала будет блокироваться до тех пор, пока канал перестанет быть пустым. Ещё примеры вычислителей (каждый вычислитель записан в кавычках, которые не являются частью вычислителя): «a?», «x := a?», «a!1».

Вычислители + каналы + общие переменные образуют *среду вычислений*. *Результат работы* вычислителя – есть результат его вычисления + новое состояние среды вычислений (состояние каналов + значения общих переменных).

Вычислитель может описываться последовательностью других вычислителей. Внутренние вычислители соединяются в нужном порядке с помощью знака «;» (без кавычек) и выполняются слева-направо, от первого к последнему. Тип вычислителя-последовательности есть тип последнего вычислителя в последовательности. Типы остальных вычислителей в последовательности должны быть **Unit**. Примеры: «x := a?; b!1», «x := (x := 2; 1)».

Вычислитель может выдавать в разное время разные результаты своей работы. Для этого в каждом вычислителе есть некий «чёртик», который по своему разумению выбирает из нескольких известных ему вариантов один. У каждого чёртика есть только конечный строго фиксированный набор вариантов. Они разделяются через *комбинатор неуправляемого выбора* (*неуправляемого* – потому что чёртик не поддаётся управлению). Обозначение в графической нотации: $e1 \bigsqcap e2$, обозначение в текстовой нотации: $e1 \bigsqcap e2$. Такое поведение вычислителя называется *недерминированным*. Примеры вычислителей с недетерминированным поведением: «$1 \bigsqcap 2$», «$x := 1 \bigsqcap x := 2$», «$x := (1 \bigsqcap 2)$», «$a!1 \bigsqcap b!3$», «$a!(b?)$».

6.2 Параллельная работа вычислителей

Вычислители могут взаимодействовать через каналы: один вычислитель посылает в канал нечто, другой вычислитель это нечто считывает. Но для этого вычислители должны работать *параллельно*. Параллельно работающие вычислители как раз и образуют среду вычислений. Для такого выполнения есть соответствующий синтаксис: e1 || e2 || e3 (это три параллельно работающих вычислителя). Комбинатор общего паралелльного выполнения («||») одинакого записывается как в графической, так и в текстовой нотации. Вычислитель, записанный таким образом, завершает свою работу только тогда, когда завершат свою работу все участники параллельной работы. Результаты всех вычислителей должны иметь тип **Unit**. В качестве состояния среды вычислений после работы параллельного вычислителя выбирается состояние после работы произвольного участника (во время параллельной работы все они работают **с копией** среды вычислений).

Стоит обратить внимание на поведение такого вычислителя: x := a? || a ! 1. Параллельно начинают выполняться оба участника параллельного

выполнения: x := a? , который хочет считать из канала «a» и записать считанное в переменную «x», и a ! 1, который хочет записать в канал «a» число 1. Но при этом в среде **могут быть и другие желающие** записать и считать в/из канала «a». Поэтому не всегда такое выполнение приведёт к тому, что в «x» будет записана 1. Может быть записана и 2, например, в таком выражении: (x := a? || a ! 1) || a ! 2.

Таким образом, оператор || не исключает вмешательства в работу вычислителей. Если необходимо оградить параллельную работу вычислителей от «чужого параллельного влияния», применяется другой комбинатор – *комбинатор параллелизма со взаимной блокировкой*, или *interlock-комбинатор*. Записывается он в графической нотации так: $e1 \between e2$, а в текстовой нотации так: $e1 \ ++ \ e2$.

Некоторые вычислители записываются в виде ключевого слова. Таковыми в языке RSL являются:

1. **skip** – пустой вычислитель – ничего не делает, сразу же завершается

2. **stop** – deadlock – вычислитель, пребывающий в состоянии «мёртвого блока»

3. **chaos** – зацикливающийся вычислитель (почти «живой блок» сам с собой)

4. **swap** – вычислитель с неспецифицированным поведением

В начале рассказа я всё-таки немного утаил правду. Чёртик бывает и управляемый. Но только управляемый извне вычислителя – другим вычислителем. Поведение вычислителя с управляемым чёртиком записывается в графической нотации так: $e1 \parallel e2$, а в текстовой нотации так: e1 [] e2. Вычислитель, работающий в параллели с таким вычислителем, может приказать управляемому чёртику не выдавать неудобные ему (не чёртику, естественно) варианты. Например, $(x := a? \parallel a!1) \between a!2$. Внешний вычислитель a!2 запретит чёртику выбирать вариант a!1, поскольку $a!1 \between a!2$ то же, что и **stop**, т.е. приводит к мёртвой блокировке, ведь это параллелизм со взаимной блокировкой – он завершится, когда завершатся все участники, но сделать они этого не могут, потому что занят канал «a», а когда туда запишет один из вычислителей, второй записать не сможет. И освободить канал никто не сможет – параллелизм же со взаимной блокировкой !

Два вычислителя называются *эквивалентными*, если на одинаковые входные данные они отвечают одинаковыми результатами работы. Пример: $x := (z := 1; 2) \equiv z := 1; x := 2$.

6.3 Основные эквивалентные преобразования

1. $e \,[\!]\, \mathbf{stop} \equiv e$
2. $e || \, \mathbf{skip} \equiv e$
3. $e \between \mathbf{skip} \equiv e$
4. $e \,[\!]\, \mathbf{chaos} \equiv \mathbf{chaos}$
5. $e \sqcap \mathbf{chaos} \equiv \mathbf{chaos}$
6. $e || \mathbf{chaos} \equiv \mathbf{chaos}$
7. $e \between \mathbf{chaos} \equiv \mathbf{chaos}$
8. $e \,[\!]\, e \equiv e$
9. $e \sqcap e \equiv e$
10. $e1 \,[\!]\, (e2 \sqcap e3) \equiv (e1 \,[\!]\, e2) \sqcap (e1 \,[\!]\, e3)$
11. $e1 || (e2 \sqcap e3) \equiv (e1 || e2) \sqcap (e1 || e3)$
12. $e1 \between (e2 \sqcap e3) \equiv (e1 \between e2) \sqcap (e1 \between e3)$
13. $(e2 \sqcap e3); e1 \equiv (e2; e1) \sqcap (e3; e1)$
14. $x := c? \between c!e \equiv x := e$
15. $(c1!e1 \between c2!e2) \equiv \mathbf{stop}$
16. $(x := c1? \between x := c2?) \equiv \mathbf{stop}$
17. $(x := c1? \,[\!]\, c2!e2) \between c1!e1 \equiv x := e1$
18. $(e1 \sqcap e2) \between e3 \equiv (e1 \between e3) \sqcap (e2 \between e3)$
19. $(e1 \sqcap e2) || e3 \equiv (e1 || e3) \sqcap (e2 || e3)$
20. $x := c? || c!e \equiv (x := e) \sqcap ((x := c?; c!e) \,[\!]\, (c!e; x := c?) \,[\!]\, (x := e))$
21. $c1 \neq c2 \Rightarrow$
    $(x := c1? || c2!e \equiv (x := c1?; c2!e \,[\!]\, c2!e; x := c1?))$

6.4 Рекомендации при упрощении выражений с параллелизмом

1. Вынести «наружу» последовательный комбинатор (например, $x := (a!1; 2)$ преобразовать в $a!1; x := 2$)
2. Отдельно упростить все выражения с interlock-комбинатором
3. При упрощении выражения - параллельной комбинации - рассмотреть все возможные варианты первого взаимодействия (обмена сообщением по каналу) и все такие варианты поместить в недетерминированный выбор
4. Не забывать, что в условном операторе (или операторе **case**) сначала необходимо вычислить значение условия и только потом приступать к вычислению одной из его ветвей, аналогичное утверждение не надо забывать про арифметические операторы

6.5 Задачи

I Выполнить эквивалентные преобразования над следующими выражениями

Считать, что все возможные взаимодействия произошли (т.е. в ответе не должно оставаться подвыражений, в которых возможно взаимодействие, например, $x := a?||a!1$).

1. $a!(5 + a?)$
   $||((x := (x := a?; 1)) \between (b!4||x := a?||b!6||a!3||y := b?))$
2. $a!(5 + (a!5; 6))$
   $||((x := (x := a?; 1)) \between (x := a?||b!6||a!3||y := b?))$
3. $(x := (a? \prod b?))$
   $||((x := (\ \textbf{if true} \prod \textbf{false then}\ b!1; 1\ \textbf{else}\ x := a?; 6\ \textbf{end}\ )) \between (b!4||a!3||y := b?))$
4. **case** $(a? \prod b?)$ **of**
   $0 \to x := a? + 1,$
   $2 \to x := b?,$
   $3 \to y := a? + 3,$
   $4 \to y := b? + a?$

end
$||a!(b?)||a!0||b!(a?+4)$

5. **case** $(a?)$ **of**
   $0 \to x := a?+1,$
   $2 \to x := (b? \sqcap a?),$
   $3 \to y := a?+3,$
   $4 \to y := b?+a?$
   end
   $||a!0||b!(a?)||(a!2 \sqcap a!3)$

6. $x := a?||$
   case $(a?+b?)$ **of**
   $1 \to x := 1,$
   $2 \to x := b?,$
   $3 \to y := 3,$
   $4 \to y := b?+a?,$
   $5 \to$ **skip**
   end
   $||a!1||b!(a?+2)||a!3$

7. **case** $(a?+a?)$ **of**
   $1 \to x := a?+1,$
   $2 \to$ **skip**,
   $3 \to y := 3,$
   $4 \to y := b?,$
   $5 \to x := (y := a?); y$
   end
   $||a!(1 \sqcap b?)||a!1||b!2||a!3||b!1$

8. $a!(x=y)$
   $||(a!\textbf{true} \not\parallel$
   $(b!4||y := x||(x := (2+b?); (a!(x=y)))||$
   $x :=$ **if** $a? \wedge a?$ **then** $6+x$ **else** 0 **end**
   $))$
   $||a!(x \geqslant y)$

9. $a!$**false**
   $||a!(4=b?)$
   $||((x := (\ \textbf{if}\ a?\ \textbf{then}\ x := b?; 2\ \textbf{else}\ b!6; x := 3; 5\ \textbf{end}\ )+x) \not\parallel (a!(\textbf{true} \sqcap \textbf{false}); (y := b?||b!1))$

Глава 7

Задачи коллоквиума №1

7.1 Вариантные определения

Enumeration

```
type
  Colour == black | white
```

это сокращение для:

```
type Colour
value
    black : Colour,
    white : Colour
axiom
   [disjoint] /* нет совпадающих элементов */
     black ~= white,
   [induction] /* для док-ва свойств про Colour по индукции */
      all p : Colour -> Bool :-
        (p(black) /\ p(white)) => (all cx:Colour :- p(cx))
```

Record constructors

```
type Collection == empty | add(Elem, Collection)
```

это сокращение для:

```
type
   Collection
value
```

```
   empty : Collection,
   add : Elem >< Collection -> Collection
axiom
   [disjoint]
   /* разные композиции вернут разные результаты */
      all e : Elem, c : Collection :- empty ~= add(e,c),
   [induction]
   /* тип Collection создается только с помощью add */
      all p : Collection -> Bool :-
            p(empty) /\
            ( all e : Elem, c : Collection :-
                  p(c) => p(add(e, c)) )
            =>
                (all c: Collection :- p(c))
```

Destructors

```
type List == empty | add(head : Elem, tail : List)
```

это сокращение для:

```
type
   List
value
    empty : List,
    add : Elem >< List -> List,
    head : List -~-> Elem,
    tail : List -~-> List
axiom
  [disjont]
    all e : Elem, l : List :- empty ~= add(e,l),
  [induction]
   ............
  [get_head]
    all e : Elem, l : List :- head(add(e, l)) is e,
  [get_tail]
    all e : Elem, l : List :- tail (add(e, l)) is l
```

Reconstructors

```
type List == empty | add(Elem <-> replace_head, List)
```

в дополнение к приведенным выше определениям вводятся следующие определения:

```
value
   replace_head : Elem >< List -~-> List
axiom
  [replace_head]
    /*голова списка становится новой, хвост не меняется*/
    all e,e1 : Elem, l : List :-
        replace_head(e, add(e1, l)) is add(e, l)
```

7.2 Задачи

I Дано explicit определение функции. Написать implicit-спецификацию функции эквивалентной данной.

1. ```
   value
         f : Int -> Nat
         f(c) is if c > 0 then c else 0-c end
   ```

2. ```
   value
         f: Int -> Real
         f(n) is
             if n = 0 then 0.0
             elsif n < 0 then f(n + 1) - 1.0
             else f(n - 1) + 1.0
             end
   ```

3. ```
   value
         f: Real -> Int
         f(x) is
             if x < 0 then 0 - f(0 - x)
             elsif x >= 1 then f(x - 1.0) + 1
             else 0.0
             end
   ```

4. ```
   value
         f: Int >< Int -~-> Int
         f(x, y) is
             if y = 0 then 1 else x * f(x, y - 1) end
         pre ~ (y < 0 \/ (x = 0 /\ y = 0))
   ```

5.
```
value
      f : Int ><Int >< Int -> write x Int >< Int
      f(a, b, c) is
          (
              if a+b > c then c else a+(x:=b; b) end,
              a * b * (if c>0 then c else 0-c end)
          )
```

6.
```
variable x, y : Int
value f : Int >< Int >< Bool -> write x read y Nat>< Bool
f (a, b, c) is
      local variable n: Int in
          if c
              then x := x + a + b; n := x**2 + y**2
              else n :=
                  if x>y then x**2 - y**2 else y**2 - x**2 end
          end  ;
          if x>y then x:=y end ;
          (n, x = y)
      end
```

7.
```
variable x, y : Int
value
      f : Int >< Nat >< Bool -> write x, y Nat >< Bool
      f (a, b, c) is
          if x>y then x := x**2 - y**2 end;
          (local variable n: Int in
              if c
                  then n := x**2 + y**2
                  else n := (x + y)**2; y := n
              end  ;
              n
           end,
          x ~= y)
```

8.
```
variable x, y : Int
value
      f : Int >< Nat >< Bool -> write x, y Nat >< Bool
      f (a, b, c) is
          if x>y then y:=x**2 end;
          (local variable n: Int in
              if c
```

```
                then n := y**2
                else n := (x + y)**2; x := n
            end  ;
            n
     end ,
    x ~= y)
```

9.
```
variable x, y : Int
value
    f : Int >< Nat >< Bool -> write x, y Nat >< Bool
    f (a, b, c) is
        let xx = x in
            if x>y then y := x**2 end;
            (local variable n: Int in
                if c
                    then n := y**2
                    else n := (x + y)**2; x := n
                end ;
                n
             end,
            xx~=y)
    end
```

10.
```
variable x, y : Int
value
    f : Int >< Int >< Bool -> write x read y Nat >< Bool
    f (a, b, c) is
        local variable n: Int in
            if c
                then x := x + a + b;n := x**2 + y**2
                else n := if x>y
                            then x**2 - y**2
                            else y**2-x**2
                          end
            end  ;
            if x>y then x:=y end ;
            (n, x=y)
       end
```

11.
```
variable x, y : Int
value
    f : Int >< Nat >< Bool -> write x, y Nat >< Bool
```

```
    f (a, b, c) is
        if x>y then x := x**2 - y**2 end;
        (local variable n: Int in
            if c
                then n := x**2 + y**2
                else n := (x + y)**2; y := n
            end  ;
            n
         end,
         x ~= y)
```

12.
```
variable x, y : Int
value
    f : Int >< Nat >< Bool -> write x, y Nat >< Bool
    f (a, b, c) is
        if x > y then y := x**2 end;
        (local variable n: Int in
            if c
                then n := y**2
                else n := (x + y)**2; x := n
            end  ;
            n
         end ,
         x ~= y)
```

13.
```
variable x, y : Int
value
    f : Int >< Nat >< Bool -> write x, y Nat >< Bool
    f(a, b, c) is
        let xx = x in
            if x>y then y:=x**2 end;
            (local variable n: Int in
                if c
                    then n:= y**2
                    else n:= ( x+ y)**2; x := n
                end ;
                n
             end,
             xx ~= y)
        end
```

14.
```
value
```

```
f : Int ><Int >< Int -> write x, y Int >< Int
f(a,b,c) is
    if a = b then
        (
            if a+b > c then c else a+b end,
            b * (if c > 0 then x := c; c else 0-c end)
        )
    else (y:=a+b; y, x:=b; a-b)
    end
```

15.
```
value
    f : Int >< Int -> write x, y Int >< Int >< Int
    f (a, b) is
        local variable v : Int := 0 in
            x:= x + a;
            for i in <.a..b.> do
                v := v + (x:=v; 2)*i
            end;
            (a,b,v)
        end
```

16.
```
value
    f : Int >< Int -> write x, y Int >< Int >< Int
    f(a, b) is
        local variable v : Int := 0 in
            y := x + a;
            for i in <.a..b.> do
                v := v + (x:=v; 2) * (x:=i; i)
            end;
            (a, b, v)
        end
```

17.
```
value
    f : Int ><Int >< Int -> write x, y Int >< Int
    f(a, b, c) is
        if a = b then
        (
            if a+b > c then c else a+b end,
            b * (if c>0 then x := c; c
                 else local ly : Int in ly:=y; y:=0-c; ly end
                 end)
        )
```

```
        else (a+b, x:=b; a-b)
        end
```

18.
```
value
    f : Int ><Int >< Int -> write x, y Int >< Int
    f(a, b, c) is
        if a = b then
            y:=x;
            (
                if a > b then c else a+b end,
                a * (if c>0 then c else x:=c; 0-c end)
            )
        else (y:=a+b; y, a-b)
        end
```

II Дана implicit спецификация функции, описать функцию эквивалентную данной в explicit форме. Специфицировать слабейшие предусловия.

1.
```
value
    f: Int -> Nat
    f(n) as m
    post
        if n < 0 then n + m = 0 else n = m end
```

2.
```
value
    gets2: Unit -> Nat
    gets2() as x
    post
        x * x = 4
```

3.
```
value
    f : Real >< Real -~-> write x Real
    f(a, b) as y
post    x + y = 1.0 /\ x**2 + y**2 = a - b
```

III Дать explicit или implicit определение функций (включая слабейшие предусловия), отвечающих требованиям аксиом

1.
```
value
```

```
    create: Unit -> C,
    add : A >< A >< C -> C,
    invert : C -~-> C,
    get : A >< C -~-> A
axiom
    all a1, a2 : A, c : C :-
        invert(add(a2, a1, c)) is add(a1, a2, invert(c))

    all a1, a2 : A, c : C :-
        invert(create()) is create()

    all a1, a2, a3 : A, c : C :-
        get(a1, add(a2, a3, c)) is
            if a1 = a2 then a3 else get(a1, c) end
```

2.
```
value
    empty : Q,
    put : Q >< E -> Q,
    toppop : Q -~-> E,
    bottompop : Q -~-> E
axiom
    (all q : Q, e : E :-
        toppop (put(q,e)) is
            if q = empty then e
            elsif e >= 0 then e
            else toppop (q)
            end
    ),

    (all q : Q, e : E :-
        bottompop (put(q,e)) is
            if q = empty then e
            else bottompop(q)
            end
    )
```

3.
```
type S, E
value
    create : Unit -> S,
    add : E >< S -> S,
    del : S -~-> S,
```

```
    next : S  -~-> S,
    get : S -~-> E,
    append : S >< S -> S
axiom
    all e : E, s : S :-
        del( add( e, s ) ) is s,

    all s : S :-
        append (create(), s) is s,

    all e : E, s1, s2 : S :-
        append (add(e,s1), s2) is add(e, append(s1,s2)),

    next( create()  ) is create(),

    all e : E, s : S :-
        next( add(e, s) ) is  append ( s, add(e, create() ) ),

    all e : E, s : S :-
        get( add(e, s) ) is e
```

4.
```
value
    create: Unit -> C,
    add : A >< C -> C,
    invert : C -~-> C,
    invertable : C  -> Bool,
    get : C >< A -~-> A,
axiom
    all a1, a2 : A, c : C :-
        invert( add(a2, add(a1, c)) ) is
                add(a1, add(a2,invert (c)))
    pre invertable(c),

    all a1, a2, a3 : A, c : C :-
        get(add(a2, add(a1, c)), a3) is
            if a1 = a3 then a2 else get(c, a3) end,

    all a : A :- get(create(), a) is a,

    invertable (create()) is true,
```

```
    all a1, a2 : A, c : C :-
        invertable (add(a2,add(a1,c))) is invertable (c)
```

5.
```
type S, K, V
value
    create : Unit -> S,
    add : K >< V >< S -> S,
    del : K >< S -~-> S,
    get : K >< S -~-> V
axiom
    all k : K :-
        del( k,create( ) ) is create( ),

    all k1,k2 : K, v : V, s : S :-
        get (k1, add (k2, v, s)) is
            if k1 = k2 then v
            else get( k1, s )
            end,

    all k1,k2 : K, v : V, s : S :-
        del( k1, add( k2, v, s )) is
            if k1 = k2 then del(k1, s)
            else add(k2, v, del( k1, s ) )
            end
```

6.
```
type S, E
value
    create : Unit -> S,
    add : E >< S -> S,
    del : E>< Nat >< S -> S,
    get : E >< S -> Nat
axiom
    all e : E :- get( e, create() ) is 0,

    all e : E, n : Nat, s : S :-
        del (e, n, create() ) is create(),

    all e1,e2 : E, s : S :-
        get(e1, add (e2, s)) is
           if e1 = e2 then 1 + get( e1, s )
           else get( e1, s )
```

```
            end,

        all e1, e2 : E, n : Nat, s : S :-
            del( e1, n, add( e2, s )) is
                if n = 0 then add(e2, s)
                elsif e1 = e2 then del(e1, n - 1, s)
                else add(e2, del( e1, n, s ) )
                end
```

7.
```
type S, E
value
    create : Unit -> S,
    add : E >< S -> S,
    del : E>< Nat >< S -~-> S,
    get : E >< S -> Nat,
axiom
    all e : E, s : S :-
        get( e, create() ) is 0,

    all e1,e2 : E, s : S :-
        get ( e1, add (e2, s)) is
            if e1 = e2 then 1 + get( e1, s )
            else get( e1, s )
            end,

    all e1, e2 : E, n : Nat, s : S :-
        del( e1, n, add( e2, s )) is
            if n <= 1 then add(e2, s )
            elsif e1 = e2 then del( e1, n-1, s )
            else add(e2, del( e1, n, s ) )
            end
```

8.
```
type S, E
value
    create : Unit -> S,
    add1 : E >< S -> S,
    del : S -~-> S,
    next : S  -~-> S,
    get : S -~-> E,
    add2 : S >< S -> S
```

```
axiom
    all e : E, s : S :-
        del( add1( e, s ) ) is s,

    all s : S :-
        add2 (create(), s) is s,

    all e : E, s1, s2 : S :-
        add2 (add1(e,s1), s2) is add1(e, add2(s1,s2)),

    next( create() ) is create(),

    all e : E, s : S :-
        next( add1(e, s)) is  add2 ( s, add1(e, create()) ),

    all e : E, s : S :-
        get( add1( e, s ) ) is e
```

9.
```
type R, Key, Val
value
    empty : Unit -> R,
    put : R >< Key >< Val -> R,
    del : R >< Key >< Val -~-> R,
    get: R >< Key -~-> Val-set,
    find: R >< Key -> Bool,
axiom
    forall r : R, k1, k2 : Key, v1, v2 : Val :-
        [find_empty]
        find (empty(), k1) is false,

        [find_put]
        find (put(r, k1, v1), k1) is true,

        [get _put]
        get (put(r, k1, v1), k1) is
            if find (r, k1)
                then get (r, k1) union {v1}
                else {v1}
            end,

        [del_put_1]
```

```
        del(put(r, k1, v1), k1, v1) is
            if find (r, k1) /\ v1 isin get (r, k1)
                then del(r, k1, v1)
                else r
              end,

        [put_put]
            put(put(r, k1, v1), k2, v2) is
              if (k1 = k2) /\ (v1 = v2)
                then put(r, k1, v1)
                else put(put(r, k2, v2), k1, v1)
              end
```

IV Проверить, является ли вторая спецификация уточнением первой. Если да, доказать это.

1.
```
scheme S1 = class
    type A, L
    value
        f1: A >< L -> L,
        f2: A >< L -> L,
        f3: L -> Nat
    axiom
        all a : A, b : L :-
            f3(f1(a,b)) is f3(b)+1
        pre f3(b) = 0,

        all a : A, b : L :-
            f3(f2(a,b)) is 1+f3(b)
        pre f3(b) = 0,

        all a: A, b : L :-
            f1(a, b) is f2(a, b)
        pre f3(b) = 0
end

scheme S2 = class
    type
        A,
        L = A-m->A
```

```
    value
        f1: A >< L -> L
        f1(a, b) is [a +> a] !! b,

        f2: A >< L  -> L
        f2(a, b) is
            if a ~isin dom b then [a +> a] union b else b end,

        f3  : L -> Nat
        f3(l) is card dom l
end
```

2.
```
scheme S1 = class
    type A, B
    value
        f1: B >< B  -> Bool,
        f2: A >< B  -> B
    axiom
        all a1, a2 : A, b : B :-
            f1(f2(a1, b), f2(a2, b)) is a1 = a2,

        all a : A, b : B :-
            f1(f2(a,b), b) is false
end

scheme S2 = class
    type
        B = A-list, A = Int, S = A-set
    value
        f1: B >< B  -> Bool
        f1(b1, b2) is hd b1 = hd b2,

        f2: A >< B  -> B
        f2(a, b) is <. a .> ^ b
end
```

3.
```
scheme S1 = class
    type A,B,C
    value
        f1 : A >< B -> Bool,
        f2 : A >< B -~-> C,
```

```
        f3 : C -~-> A >< B
    axiom
        all  a: A, b: B :-
            f3( f2(a, b) ) is (a,b)
        pre f1(a,b) = true
end

scheme S2 = class
    type
        A = Int, B = Nat, C = Int
    value
        f1 : A >< B -> Bool
        f1(i,n) is i * i = n,

        f2 : A >< B -~-> C
        f2(i, n) is i,

        f3 : C -~-> A >< B
        f3(cc) as (a,b)
        post cc = a /\ b = cc * cc
end
```

V Дать определение эквивалентное данному без использования вариантных возможностей RSL

1. ```
   type Collection ==
                  empty
                | add1 (Elem, Collection)
                | add2(Elem, Elem),
         Elem
   ```

2. ```
   type Collection ==
                  empty
                | add1 (Elem, tail:Collection <-> tail),
         Elem
   ```

3. ```
   type Collection ==
                  empty
                | add1 (Elem, Collection)
                | add2(head1 : Elem, Collection),
         Elem
   ```
   ```

4.
```
type Collection ==
                empty
              | add1 (head:Elem)
              | add2(head : Elem, Collection),
       Elem
```

5.
```
type Collection ==
                empty
              | add1 (head:Elem)
              | add2(head : Elem <-> head),
       Elem
```

6.
```
type Collection ==
                empty
              | add (Collection <-> head, tail:Collection)
```

7.
```
type Collection ==
                empty
              | add1 (Collection <-> tail1,
                      tail:Collection <-> tail2)
```

VI Определение типа с учетом его ограничений

В каждой задаче требуется привести определение типа SYSTEM с нужной по заданию семантикой. В определении должны быть учтены все явные и неявные ограничения на элементы данного типа. При определении типа SYSTEM можно построить определения дополнительных типов. Не разрешается использовать разделы **value** (кроме определения констант), **axiom** и **variable**.

1. *Анаграммы*. Анаграмма представляет собой пару слов, состоящих из одних и тех же букв. Каждая буква, встречающаяся в одном слове, встречается и в другом, причем ровно в том же количестве. Например, (ВЕШАЛ, ЛЕВША).

2. *Панграммы*. Панграмма – это набор слов, в которых каждая буква алфавита встречается ровно один раз. Например, АБРЕК ЗЮЙД НЭП СЪЁМ ХЛЫЩ ЦУГ ЧИЖ ШТОФ ЯВЬ.

3. *Линейный палиндром*. Палиндром (перевертыш) – слово (фраза), читающееся одинаково от первого символа к последнему и от последнего к первому. Самым длинным в мире палиндромом принято

считать финское слово KUULILENNUTEETUNNELILUUK. В английском языке самое длинное из этого типа - REDIVIDER (что-то вроде перегородки). Самым коротким русским палиндромом является слово: «О!». Пример фразы-палиндрома – «И мал Иван, а лупил у лип улана вилами!».

4. *Квадратный палиндром*. Палиндром представляется в виде квадрата из символов. В таком виде палиндром читается 4-мя способами: по горизонтальным и вертикальным рядам - слева направо и справа налево. Пример:
   S A T O R
   A R E P O
   T E N E T
   O P E R A
   R O T A S

5. *Городской метрополитен*. Модель содержит сведения о станциях (и их названиях), линиях и переходах между станциями. Учтите, что внутри линии названия станций не должны повторяться. Станции могут иметь одинаковые имена, но при этом лежать на разных линиях.

6. *Идеальная школа*. Идеальная школа состоит из учителей и школьников. Школьники сгруппированы в классы. В каждом классе не более 20 школьников. В школе имеются учителя по математике, литературе, истории, рисованию, химии, физике, физкультуре и пению. Любой учитель не может вести более 2 различных предметов. При составлении спецификации учтите большое количество предметов в школе.

7. *Общежитие*. Общежитие состоит из этажей одинаковой планировки. Из-за технических особенностей количество этажей не должно превышать 20. Этаж состоит из комнат. Есть комнаты на 2, 3 или 4 человека. Всего этаж должен вмещать не менее 100 человек. Кроме того, на каждом этаже есть 2 кухни и комната дежурной.

8. *Перекидной настенный календарь*. Учтите, что год бывает високосный и невисокосный. Номер года задать в виде константы RSL с неопределенным значением. Предусмотреть возможность хранения комментария к каждому дню («To-Do»).

9. *Комплексный обед*. В столовой решили составлять меню комплексного обеда из расчета не более 100 рублей. В меню выделили пер-

вые блюда, вторые блюда, гарниры, салаты, напитки. Комплексный обед должен содержать блюда каждого типа в количество одной позиции.

10. *Позиция в игре «Морской бой»*. Позиция представляет из себя поле размером 10 x 10. На поле располагаются корабли. Каждый корабль представляет собой линию длины от 1 до 4. Количество кораблей в сумме с длиной равно 5. Любые два корабля не должны пересекаться. Допускается расположение кораблей с соприкосновением угловых вершин. Соприкосновение граней кораблей недопустимо.

11. *Планировка района*. Район имеет размер M x N единиц. Имеется множество различных зданий, заданных длиной и шириной. Все эти здания расположены без наложений.
Дополнительное требование в качестве более сложного варианта:

    * между домами должно быть не менее K единиц длины свободного пространства

12. *Компьютерная сеть*. Компьютерная сеть – набор компьютеров. На компьютере могут быть запущены как клиентские, так и серверные программы. В сети обязательно должен быть DHCP-, DNS-, SMTP-, HTTP- сервера (машины, на которых запущены DHCP-... сервисы), причем на одной машине не может быть запущено более одного сервиса. Сеть построена на основе топологии:

    - линия
    - иерархической
    - кольцо
    - звезда

Приложение A

Графическая и текстовая нотации

текст	граф	текст	граф	текст	граф	текст	граф
is	$\equiv$	~=	$\neq$	->	$\rightarrow$	-~->	$\widetilde{\rightarrow}$
:-	$\bullet$	**	$\uparrow$	=>	$\Rightarrow$	><	$\times$
/\	$\wedge$	\/	$\vee$	>=	$\geqslant$	<=	$\leqslant$
all	$\forall$	exists	$\exists$	exists!	$\exists!$	<->	$\leftrightarrow$
isin	$\in$	~isin	$\notin$	union	$\cup$	inter	$\cap$
>>	$\supset$	<<	$\subset$	>>=	$\supseteq$	<<=	$\subseteq$
-list	*	-inflist	$^\omega$	<.	$\langle$	.>	$\rangle$
-m->	$\underset{m}{\longrightarrow}$	-~m->	$\underset{m}{\widetilde{\longrightarrow}}$	+>	$\mapsto$	!!	$\dagger$
||	$\parallel$	++	$\nparallel$	|=|	$\llbracket$	|^|	$\prod$
#	$\circ$	`alpha	α	`beta	β	`gamma	γ

Приложение B

Приоритет операций в RSL

Операции с большим приоритет обрабатываются первее.

Приоритет операций в типовых выражениях

приоритет	операторы	ассоциативность
1	,	правая
2	$\xrightarrow[m]{}$ $\widetilde{\xrightarrow[m]{}}$	правая
3	$\times$	
4	**-set -infset** * $^{\omega}$	

Приоритет операций в выражениях со значением

приоритет	операторы	ассоциативность
1	$\forall \exists \exists!$	правая
2	$\equiv$	
3	$[\!] \sqcap \| \nparallel$	правая
4	;	правая
5	$:=$	
6	$\Rightarrow$	правая
7	$\vee$	правая
8	$\wedge$	правая
9	$=\neq><\geqslant\leqslant\subset\supset\subseteq\supseteq\in\notin$	
10	$+ - \setminus \hat{} \cup \dagger$	левая
11	$*/ \circ \cap$	левая
12	$\uparrow$	
13	$\sim$ префиксы суффиксы	

Литература

[1] *Кузьменкова . ., Петренко . .* Формальная спецификация программ на языке RSL. Методическое пособие по практикуму. — М.: ВМиК, 1999. — С. 57.

[2] *Кузьменкова . ., Петренко . .* Формальная спецификация программ на языке RSL. Конспект лекций. — М.: ВМиК, 2001. — С. 106.

[3] RAISE course института UNU-IIST. — ftp://ftp.iist.unu.edu/pub/RAISE/course_material/.

[4] The Raise Specification Language (Bcs Practitioner Series). — Prentice Hall, 1992. — P. 106.

[5] RSL Type Checker отдела Технологий программирования ИСП РАН. — http://www.ispras.ru/∼RedVerst/RedVerst/Lectures and training courses/MSU course Formal specification of software/RSL2CPP/.

[6] RSL Type Checker института UNU-IIST. — ftp://ftp.iist.unu.edu/pub/RAISE/dist/.