

Операции над треками

Егор Горбунов

16 апреля 2016 г.

Описание операций и языка

Основные операции

- Арифметические операции

Вариант №1: $\alpha \in \mathbb{R}, X \in \mathbb{R}^n, Y \in \mathbb{R}^n$. Кодомен всех операций — \mathbb{R}^n .

- $X \diamond Y := (x_1 \diamond y_1, x_2 \diamond y_2, \dots, x_n \diamond y_n), \diamond \in \{+, -, \cdot, /\}$
- $\alpha \diamond Y := (\alpha \diamond y_1, \alpha \diamond y_2, \dots, \alpha \diamond y_n), \diamond \in \{^{\wedge}, +, -, \cdot, /\}$
- $X \diamond \alpha := (x_1 \diamond \alpha, x_2 \diamond \alpha, \dots, x_n \diamond \alpha), \diamond \in \{^{\wedge}, +, -, \cdot, /\}$

Тут возникает вопрос о том, что делать с отрицательными значениями: имеет ли смысл их отрисывать в браузере? имеют ли вообще смысл операция «-» и домножение на отрицательное число? Мне, кажется, что они не нужны и вполне заменяются предикатами и условным оператором, описанными ниже.

Вариант №2: Всё то же, что и выше, но убираем операцию «-» и теперь $\alpha \in \mathbb{R}_+$, а $X, Y \in \mathbb{R}_+^n$

Вариант №3: операции все, что в 1-ом варианте, но: $x \diamond y \rightarrow \max(0, x \diamond y)$

Тут нужно решить, но я склоняюсь ко 2-ому варианту.

- Предикаты Тут всё понятно. Пусть есть $X \in \mathbb{R}^n, Y \in \mathbb{R}^n$, тогда:

$$\begin{aligned} \diamond &: (\mathbb{R}^n, \mathbb{R}^n) \rightarrow \{0, 1\}^n \\ X \diamond Y &= (x_1 \diamond y_1, x_2 \diamond y_2, \dots, x_n \diamond y_n) \\ \diamond &\in \{<, >, \leq, \geq, =, \neq\} \end{aligned}$$

- Связки на предикатах Опять же, ничего необычного: $A, B \in \{0, 1\}^n$, тогда определяем:

$$A \text{ OR } B, A \text{ AND } B, \text{ NOT } A \in \{0, 1\}^n$$

- Условный оператор Пускай теперь $P \in \{0, 1\}^n, X \in \mathbb{R}^n, Y \in \mathbb{R}^n, \alpha \in \mathbb{R}, b \in \mathbb{R}$, тогда полезна

следующая операция:

if P **then** X **else** Y := P · X + (1 – P) · Y

if P **then** a **else** Y := P · a + (1 – P) · Y

if P **then** X **else** b := P · X + (1 – P) · b

if P **then** a **else** b := P · a + (1 – P) · b

С помощью этой операции легко выражаются всевозможные фильтры.

PS: тут можно поспорить про синтаксис.

- **Псевдонимы для треков** Названия треков довольно длинные, поэтому полезно будет в запросах уметь ссылаться на конкретные треки через псевдонимы. Предлагаю описывать псевдонимы в самом начале запроса (как минимум потому, что autocompletion сможет их подхватить) так:

$x = \text{longTrackName1}, \dots, z = \text{longTrackNameK}$ **in** Expression(x, \dots, z)

- **Имя для нового трека** Результатом каждого запроса Q является новый трек, пользователь должен иметь возможность задать ему имя. Это будет делаться так:

newTrack <- Q

Если имя не указано, то оно будет сгенерировано или записано с именем lastResult, что перепишет предыдущий неименованный результат.

- **Показать отрезки, где выполнен предикат** Это может быть реализовано через условный оператор.
- **Показать трэк** Все операции генерируют трэки, но не добавляют их к отображению. Чтобы добавить какой-то трэк, который был сгенерирован нужно специальное слово, например:

show TRACK

Дополнительные операции:

- **Изменение размера корзины** Такая возможность понадобится, если пользователь захочет совершать операции над треками с разными размерами корзин. Кажется, что в силу серьёзности данных, это нужно делать аккуратно. Совсем честно мы можем изменить число корзин гистограммы (трека) с n на m только если $n > m$ и $m|n$, т.е. если текущий размер корзины делит новый размер. При этом ещё возникают некоторые проблемы на границах рассматриваемого участка. Вопрос: нужна ли возможность изменять размер корзины на произвольный? Если да, то видимо наиболее разумный способ реализации – это каждую корзину равномерно разбить на корзины раз-

мера 1, после чего уже пересчитать гистограмму для нового размера корзины.

`resize(track, bpNewBinSize)`

- **Операции на подотрезках** Назвал страшно, а речь идёт о построении новых треков посредством сканирования окном гистограммы (трека) и подсчёта в окне какой-нибудь функции: `max`, `min`, `+`, `×`, ... Синтаксис в духе:

`window(track, windowSize, Function)`

Эта штука переключается с изменением размера гистограммы, но тут `windowSize` — это число бинов (столбцов гистограммы), а не `bp`. `Function` действует из $\mathbb{R}^{\text{windowSize}}$ в \mathbb{R} и пока кажется, что можно предоставить некоторый фиксированный выбор этих возможных функций: `MEAN`, `MAX`, `MIN`, `SUM`, `MUL`, `VAR`, `MEDIAN`, `NORMALIZE`, хотя, конечно, это не выглядит гибко, но если подумать, то многие функции можно выразить через имеющуюся арифметику на треках и изменение размера корзины.

Примеры использования

```
newTrack <- if (track1 >= track2 AND track1 <= track2 + 100) then track1 else track2
```

```
newTrack <- X = track1, Y = track2 in (if X > Y then X else Y)
```

```
newTrack <- 1 / (2 ^ track1)
```

```
newTrack <- X = track1 in (if track2 > 20 then X else 0)
```

```
Y - X + (if (if Y > 0 then Y else X) < 0 then Z else 0) * Y
```

Грамматика (описание языка)

```
STATEMENT = ID <- EXPR
           | EXPR
           | show ID
EXPR = ID
      | NUM
      | (EXPR)
      | EXPR + EXPR
      | EXPR - EXPR
      | EXPR * EXPR
      | if PRED then EXPR else EXPR
PRED = TERM
      | not TERM
      | (PRED)
      | PRED or PRED
      | PRED and PRED
TERM = true | false | REL
REL = EXPR <= EXPR | EXPR >= EXPR | EXPR == EXPR | EXPR < EXPR | EXPR > EXPR
NUM = [1-9][0-9]* | 0
ID = [a-zA-Z_][0-9a-zA-Z-Z_]*
```

Преобразованная грамматика

- Главное выражение:

```
S -> %track_name% <- E
    | show %track_name%
    | E
```

- Арифметика на треках:

```
E -> E + T
    | E - T
    | T
T -> T * X
    | T / X
    | X
X -> X ^ P
    | P
P -> (E)
    | if PRED then E else E
    | %track_name%
    | %number%
```

- Предикаты:

```
PRED -> PRED or A
      | A
A -> A and B
    | B
B -> (PRED)
    | not PRED
    | true
    | false
    | R
R -> E < E
    | E > E
    | E <= E
    | E >= E
    | E == E
    | E != E
```

Грамматика без левой рекурсии

```
STATEMENT -> ID "<-" TRACK_EXPR
            | show ID
            | TRACK_EXPR

TRACK_EXPR -> E_TERM SUM_SUB

SUM_SUB -> "+" E_TERM SUM_SUB
         | "-" E_TERM SUM_SUB
         | .

E_TERM -> FACTOR MUL_DIV

MUL_DIV -> "*" FACTOR MUL_DIV
         | "/" FACTOR MUL_DIV
         | .

FACTOR -> "(" TRACK_EXPR ")"
        | if PREDICATE then TRACK_EXPR else TRACK_EXPR
        | ID
        | NUMBER

PREDICATE -> B_TERM OR_PRED

OR_PRED -> or B_TERM OR_PRED
        | .

B_TERM -> NOT_FACTOR AND_PRED

AND_PRED -> and NOT_FACTOR AND_PRED
        | .

NOT_FACTOR -> not FACTOR
            | FACTOR

FACTOR -> "(" PREDICATE ")"
        | RELATION
        | true
        | false

RELATION -> TRACK_EXPR "<=" TRACK_EXPR
          | TRACK_EXPR ">=" TRACK_EXPR
          | TRACK_EXPR "==" TRACK_EXPR
          | TRACK_EXPR "!=" TRACK_EXPR
          | TRACK_EXPR ">" TRACK_EXPR
          | TRACK_EXPR "<" TRACK_EXPR

ID -> [a-zA-Z_][a-zA-Z_0-9]*

NUM -> %float_num_regex%
```

Около-РЕГ грамматика

```
STATEMENT <- ID "<-" TRACK_EXPR
           / show ID
           / TRACK_EXPR
TRACK_EXPR <- E_TERM ("+" E_TERM / "-" E_TERM)*
E_TERM <- FACTOR ("*" FACTOR / "/" FACTOR)*
FACTOR <- "(" TRACK_EXPR ")"
          / if PREDICATE then TRACK_EXPR else TRACK_EXPR
          / ID
          / NUMBER
PREDICATE <- B_TERM OR_PRED
OR_PRED <- (or B_TERM OR_PRED)?
B_TERM <- NOT_FACTOR AND_PRED
AND_PRED <- (and NOT_FACTOR AND_PRED)?
NOT_FACTOR <- not FACTOR
           / FACTOR
FACTOR <- "(" PREDICATE ")"
          / RELATION
          / true
          / false
RELATION <- TRACK_EXPR "<=" TRACK_EXPR
          / TRACK_EXPR ">=" TRACK_EXPR
          / TRACK_EXPR "==" TRACK_EXPR
          / TRACK_EXPR "!=" TRACK_EXPR
          / TRACK_EXPR ">" TRACK_EXPR
          / TRACK_EXPR "<" TRACK_EXPR
ID <- [a-zA-Z_][a-zA-Z_0-9]*
NUM <- %float_num_regex%
```