НИУ ВШЭ – Пермь, ПГНИУ

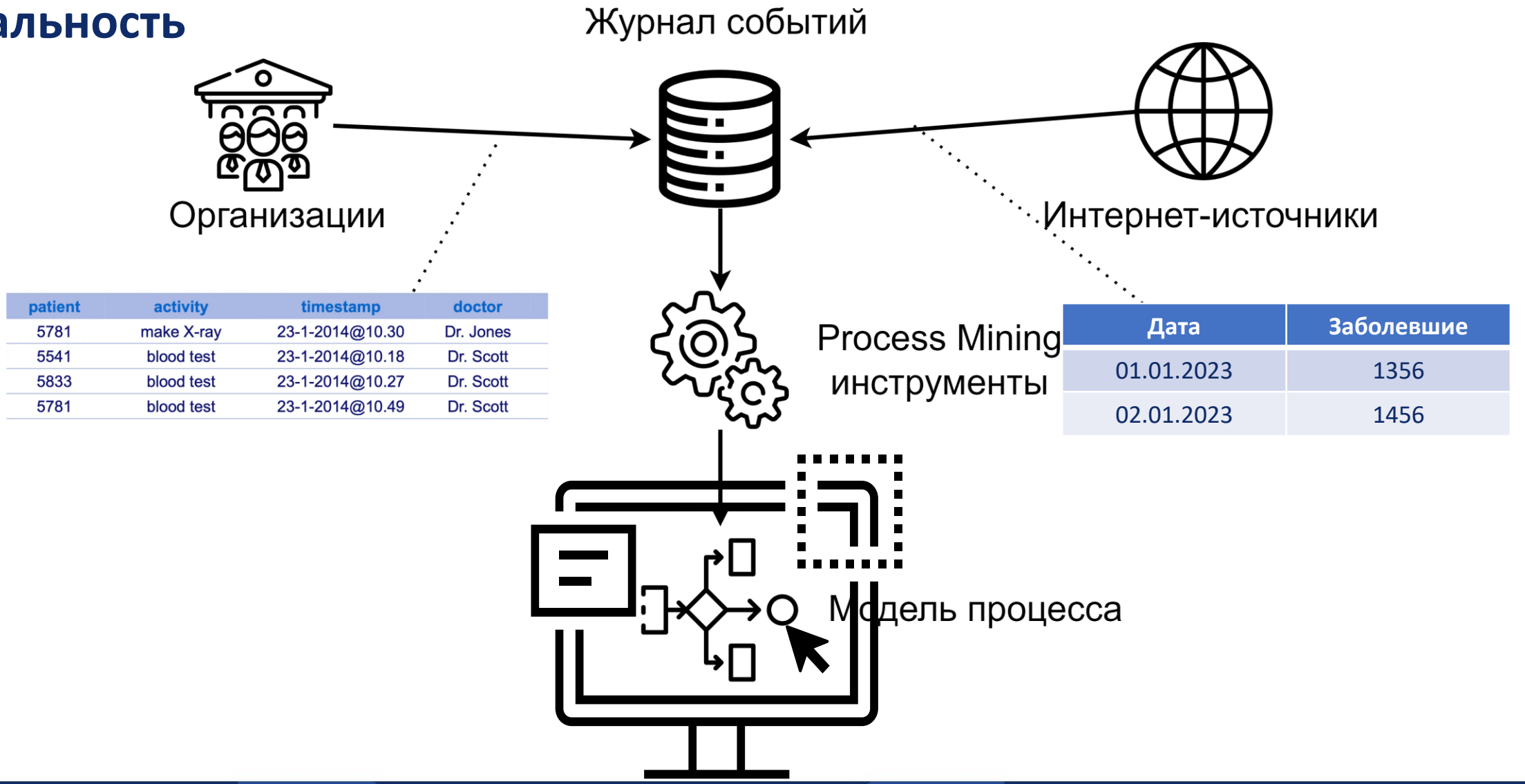
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Платунов Антон Игоревич, Лядова Людмила Николаевна

РАЗРАБОТКА КОНСТРУКТОРА ЖУРНАЛОВ СОБЫТИЙ С ДОПОЛНИТЕЛЬНЫМИ АТРИБУТАМИ

ТРИС, 2023

Актуальность



patient	activity	timestamp	doctor
5781	make X-ray	23-1-2014@10.30	Dr. Jones
5541	blood test	23-1-2014@10.18	Dr. Scott
5833	blood test	23-1-2014@10.27	Dr. Scott
5781	blood test	23-1-2014@10.49	Dr. Scott

Дата	Заболевшие
01.01.2023	1356
02.01.2023	1456

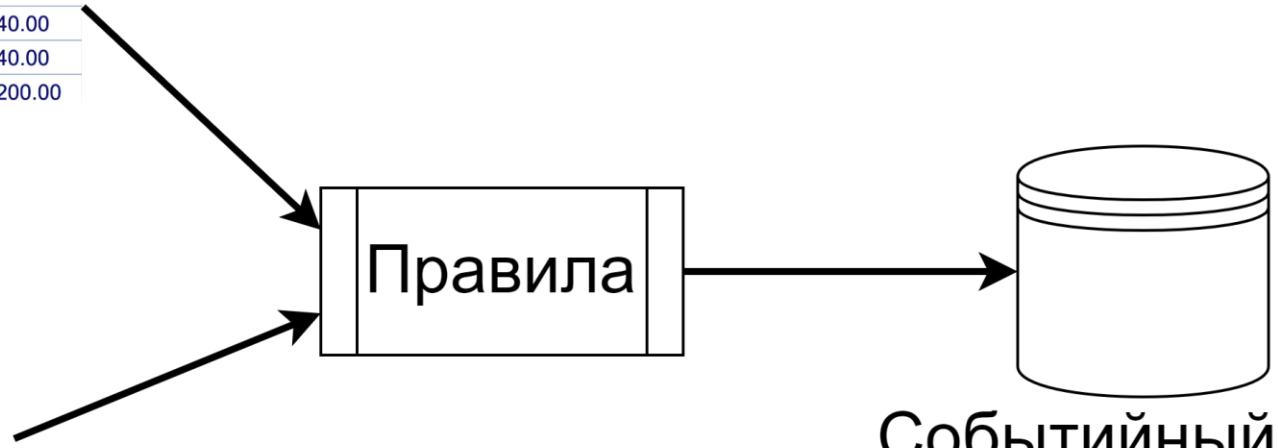
Актуальность

patient	activity	timestamp	doctor	age	cost
5781	make X-ray	23-1-2014@10.30	Dr. Jones	45	70.00
5541	blood test	23-1-2014@10.18	Dr. Scott	61	40.00
5833	blood test	23-1-2014@10.27	Dr. Scott	24	40.00
5781	blood test	23-1-2014@10.49	Dr. Scott	45	40.00
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200.00

Журнал событий

Дата	Заболевшие
01.01.2023	1356
02.01.2023	1456

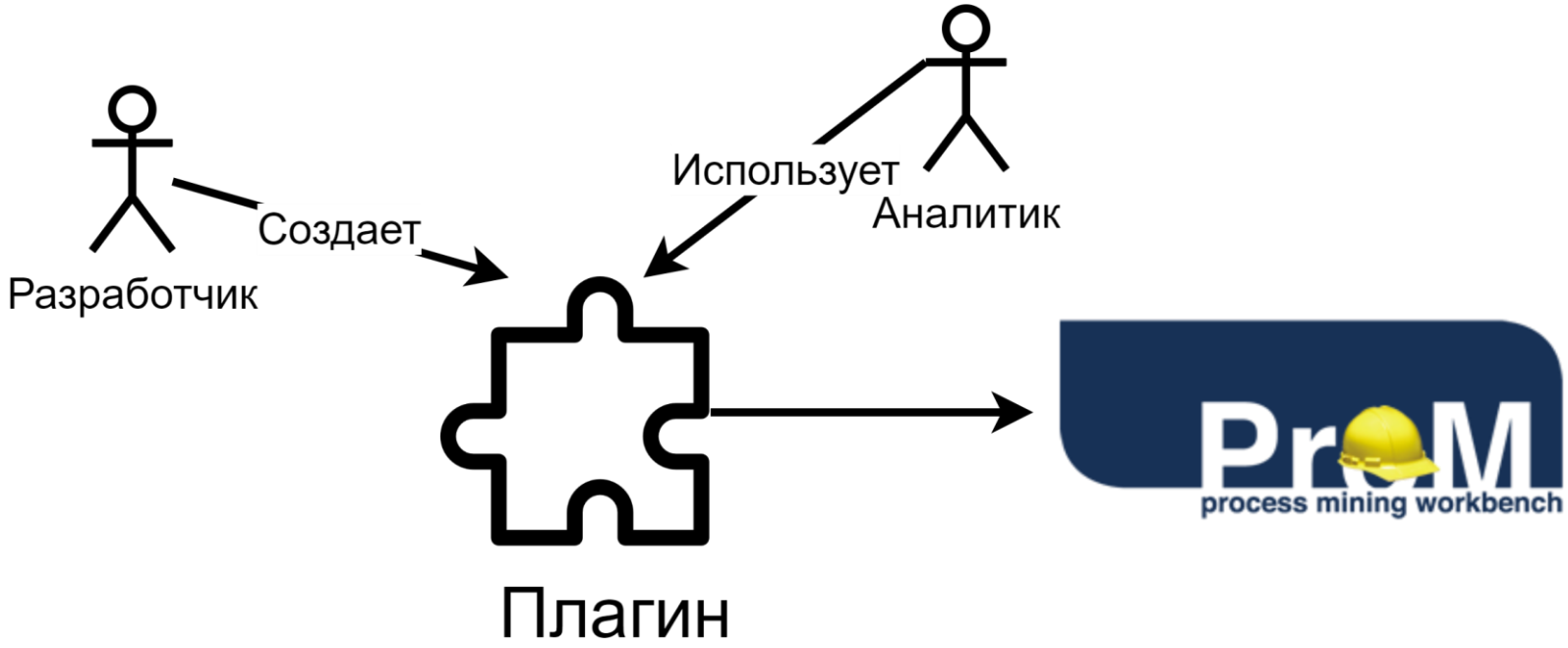
Временной ряд



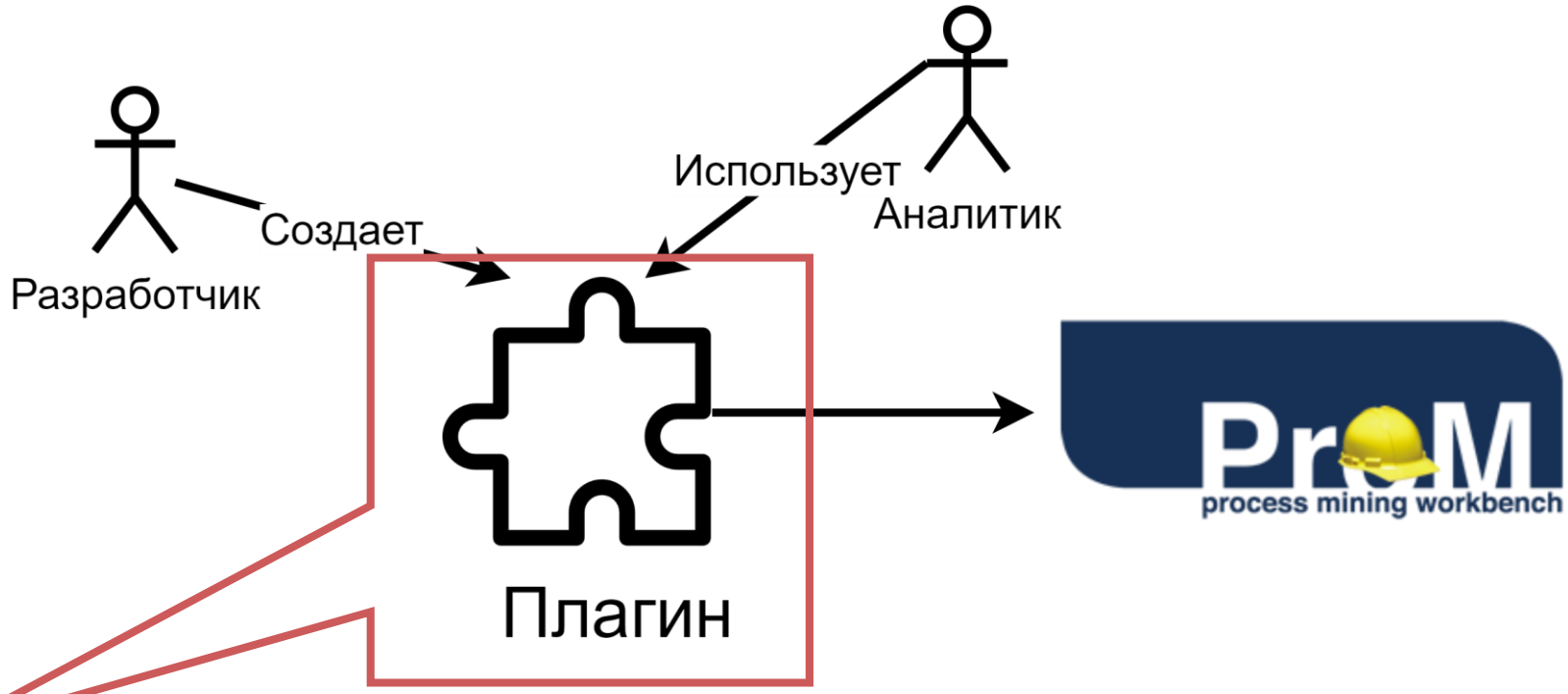
Событийный ряд

Дата	Заболевшие	Тип события
01.01.2023	1356	Высокий рост
02.01.2023	1456	Низкий рост

Актуальность



Актуальность



Проблема – отсутствие инструментов, позволяющих исследователю самостоятельно определять правила анализа событийных рядов без привлечения программиста

Объект и предмет

- ***Объект –*** процессы, представленные как последовательности событий и связанных с ними числовых показателей
- ***Предмет*** – программные средства, реализующие алгоритмы генерации и анализа событийных рядов (журналов событий, дополненных нестандартными атрибутами) с применением онтологии

Цель и задачи

- **Цель** – программная реализация в формате «low-code» программы, которая позволит проводить анализ событийных рядов с помощью задаваемых в конструкторе правил
- **Задачи**:
    1. Проведение анализа предметной области
    2. Проектирование исследовательского прототипа – конструктора правил генерации и обработки журналов событий и интерпретатора.
    3. Программная реализация исследовательского прототипа и его тестирование

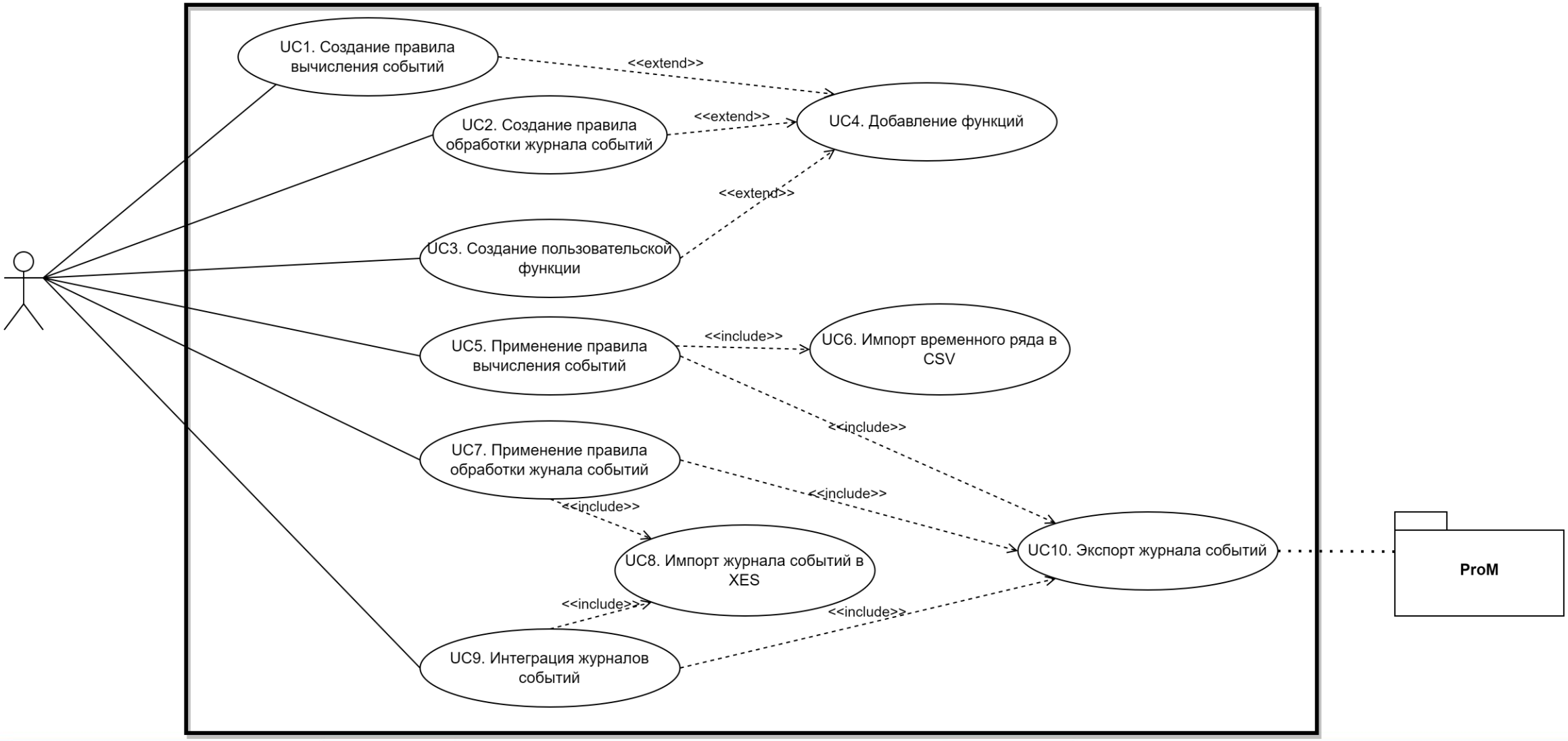
Функциональные требования

1. Определение источников данных, используемых для создания событийных рядов и их наполнения числовыми показателями.
2. Описание предметной области с помощью онтологий.
3. Описание правил определения событий на основании числовых показателей и их сохранение в онтологию в форматах журналов событий.
4. Связывание описанных в онтологии событий с числовыми характеристиками из других источников на основании заданных параметров.
5. Экспорт журналов событий для проведения анализа построенных событийных рядов с помощью плагинов ProM.

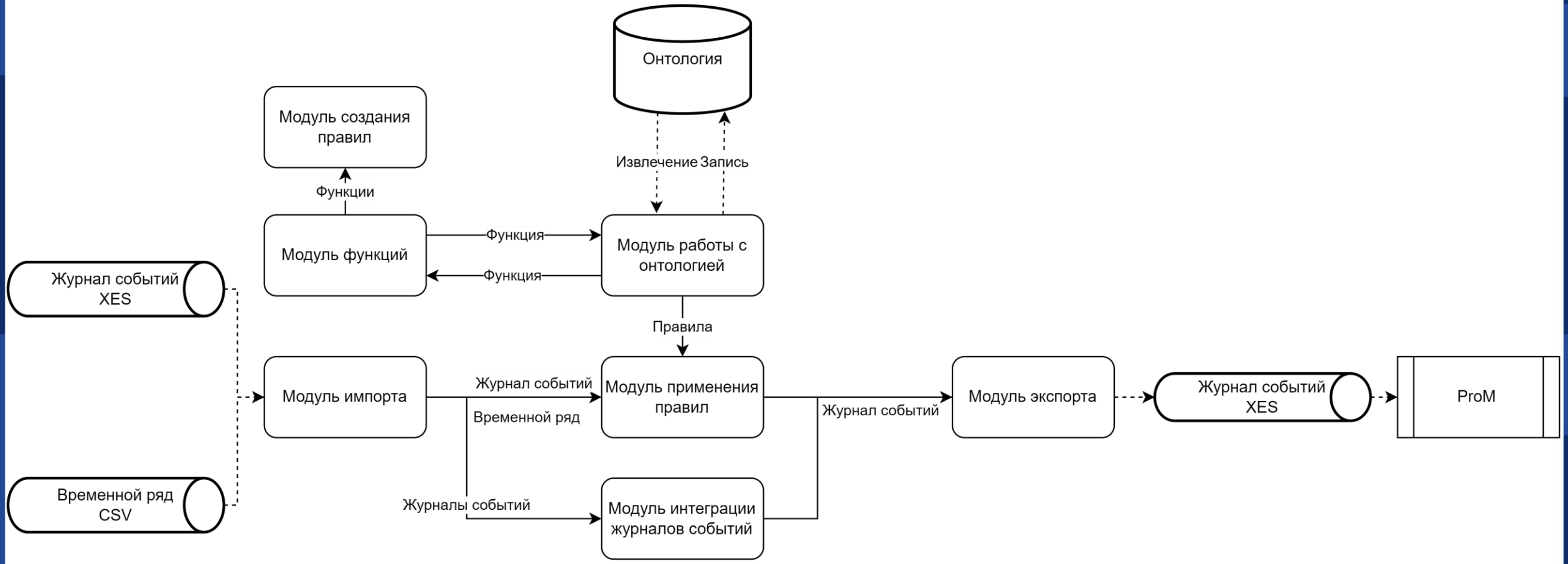
Нефункциональные требования

1. Совместимость данных с форматами данных ProM для интеграции со средствами анализа процессов (для проведения анализа построенных событийных рядов с помощью плагинов ProM).
   Структура событий должна соответствовать стандарту XES.
2. События, правила их определения и интерпретации должны храниться в онтологии.
   Для их реализации должен быть разработан формальный язык описания правил.

Функциональные требования – диаграмма прецедентов



Структура системы



Представление правил генерации и обработки журналов событий

Основа представления алгоритмов – правил генерации и обработки журналов событий – *суперпозиция функций*:

$$f = \sigma(f_0, f_1, \ldots, f_n),$$

где функция f определена как суперпозиция из $n + 1$ функции $f_0, f_1, \ldots, f_n$; функции $f_1, \ldots, f_n$ имеют свои наборы параметров для вычисления: $f_1\ (x^1_1, x^1_2, \ldots, x^1_{k1}), \ldots, f_n\ (x^n_1, x^n_2, \ldots, x^n_{kn})$, а результат вычисляется как функция

$$f\ (x^1_1, x^1_2, \ldots, x^1_{k1}, \ldots, x^n_1, x^n_2, \ldots, x^n_{kn}) = f_0\ (f_1\ (x^1_1, x^1_2, \ldots, x^1_{k1}), \ldots, f_n\ (x^n_1, x^n_2, \ldots, x^n_{kn})).$$

Все рассматриваемые функции являются *частичными*, т. е. не всюду определёнными – могут существовать такие комбинации значений аргументов, для которых значения функций не существуют: $f\ (x^1_1, x^1_2, \ldots, x^1_{k1}, \ldots, x^n_1, x^n_2, \ldots, x^n_{kn})$ не существует, если

1) не существует хотя бы одно из значений $f_1\ (x^1_1, x^1_2, \ldots, x^1_{k1}), \ldots, f_n\ (x^n_1, x^n_2, \ldots, x^n_{kn})$, или
2)эти значения существуют и равны $b_1, \ldots, b_n$, но не существует значение $f_0\ (b_1, \ldots, b_n)$.

Такое определение задаёт *способ описания алгоритма вычисления функции, вид правил*.

Представление правил генерации и обработки журналов событий

В системе имеется базовый набор операций, стандартных функций:

- операции над значениями стандартных типов (арифметические операции, операции над строками, операции сравнения и т п.) и
- встроенные функции (*MAX*, *MIN*, *AVG*, *SUM* и пр.).

Для упрощения интерфейса конструктора правил (построителя выражений) логические операции (*AND*, *OR*, *XOR*) реализованы *как функции* по аналогии с соответствующими функциями MS Excel – это также упрощает учёт приоритетов операций при формировании выражений и разборе сформированного пользователем определения функции, при интерпретации (вычислении) построенной функции.

Конструктор выражений допускает также использование функций *SWITCH* или *IFTHEN* (для определения альтернативных вариантов вычисления в зависимости от заданных условий), *FOREACH* (для обработки данных в цикле)…

Пользовательские правила строятся как суперпозиция существующих функций. Разработанные описания далее могут использоваться для создания новых правил…

Формальный язык описания правил – грамматика языка

$$G = \{ A, N, P, S \},$$

где ***A*** – *алфавит языка* – множество *терминальных символов*;

N – множество *нетерминальных символов*;

P – множество *правил грамматики*;

S – *начальный* (*целевой*) *символ*.

Разработанная грамматика языка включает:

- **56** терминальных символов (кроме букв)
- **50** нетерминальных символов и правил их определения

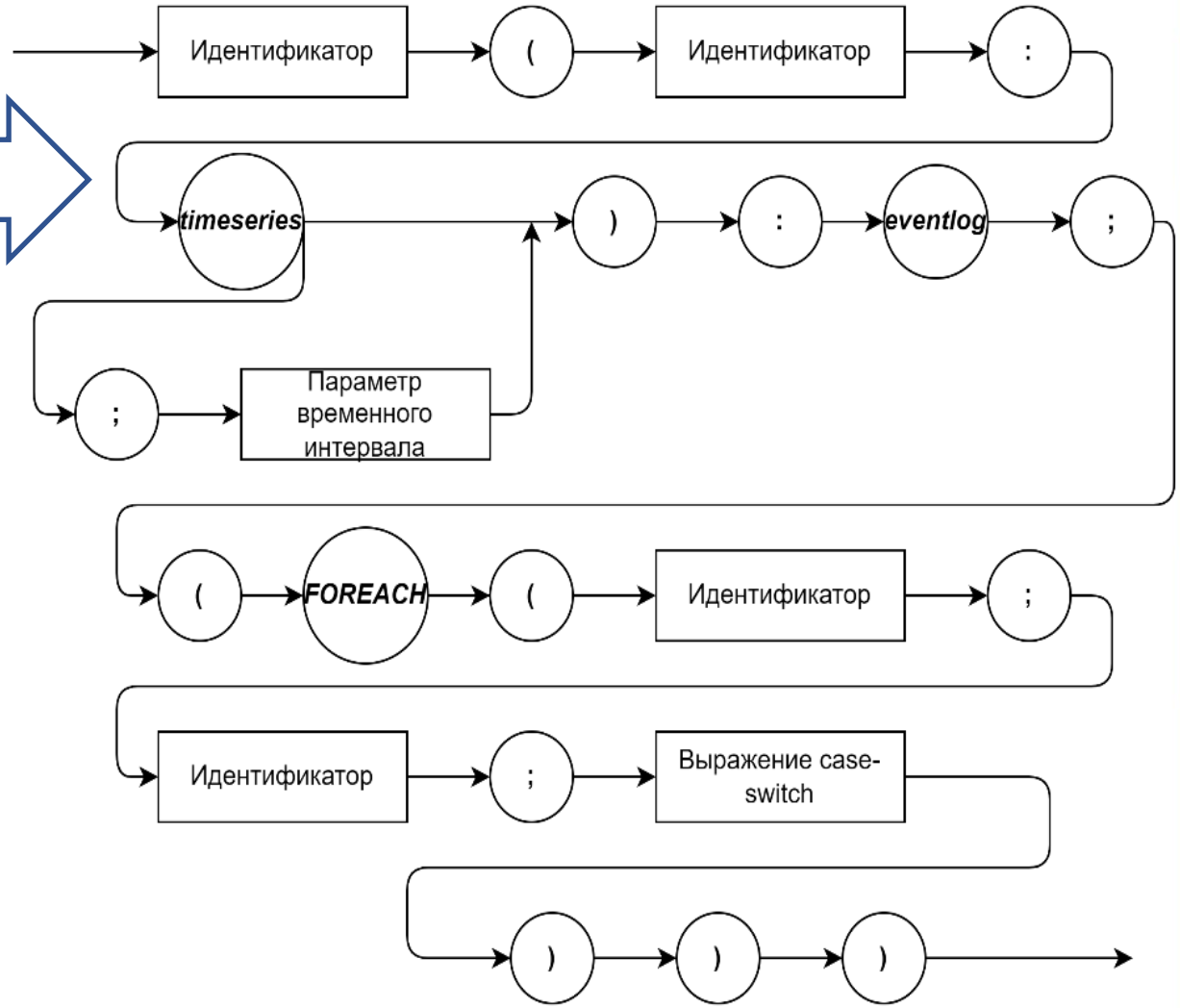
Целевой символ грамматики – функция (правило, задающее алгоритм генерации журнала или его обработки).

Грамматика относится к классу *LL*(1).

Формальный язык описания правил: пример

БНФ: → **Диаграмма Вирта**

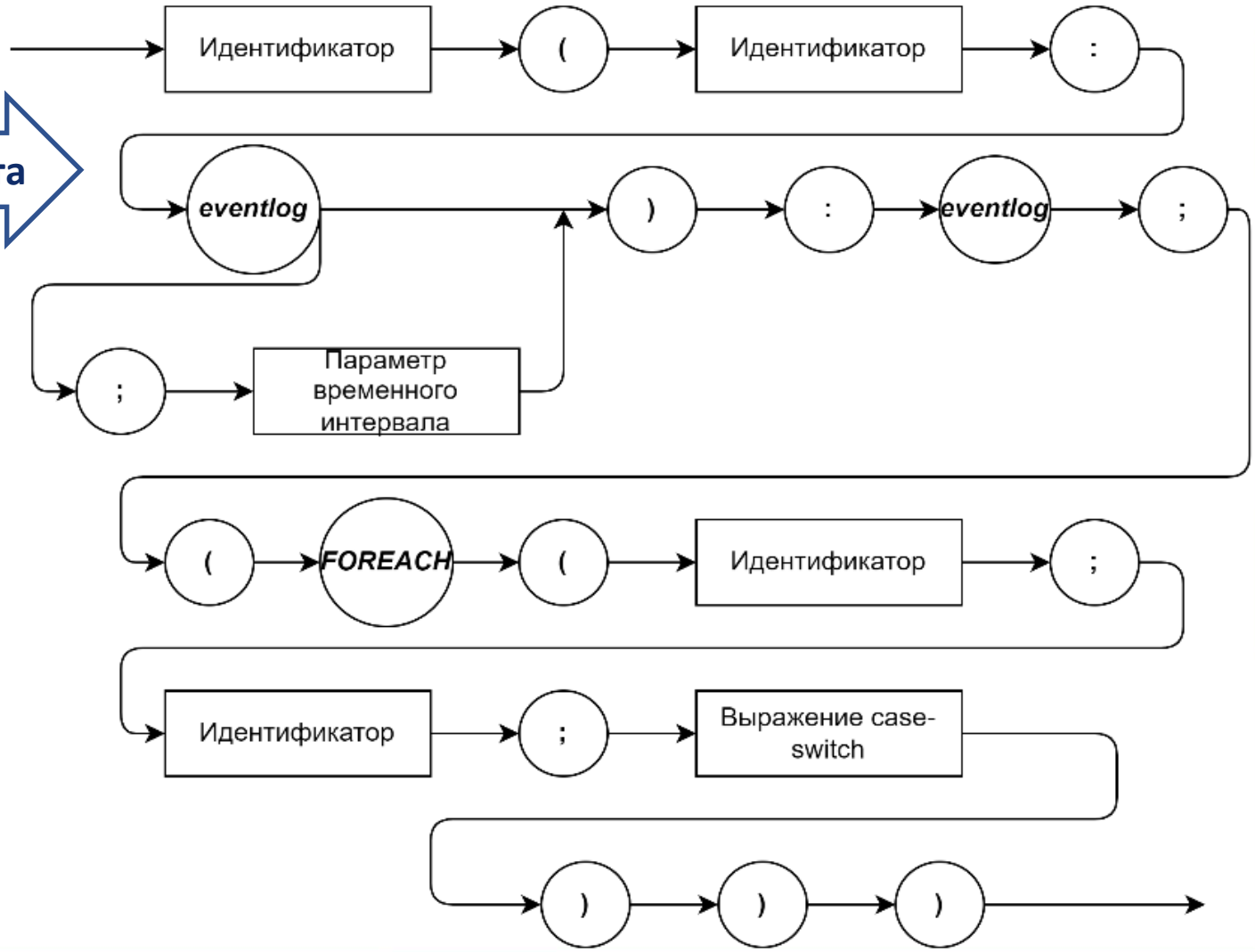
```
<правило вычисления событий> ::=
      <идентификатор>
      ( <идентификатор> : timeseries
      [ ; <параметр временного интервала> ] )
      : eventlog ;
      ( FOREACH  ( <идентификатор> ;
      <идентификатор> ;
      ( <выражение case-switch> ) ) )
```



Формальный язык описания правил: пример

БНФ: → **Диаграмма Вирта**

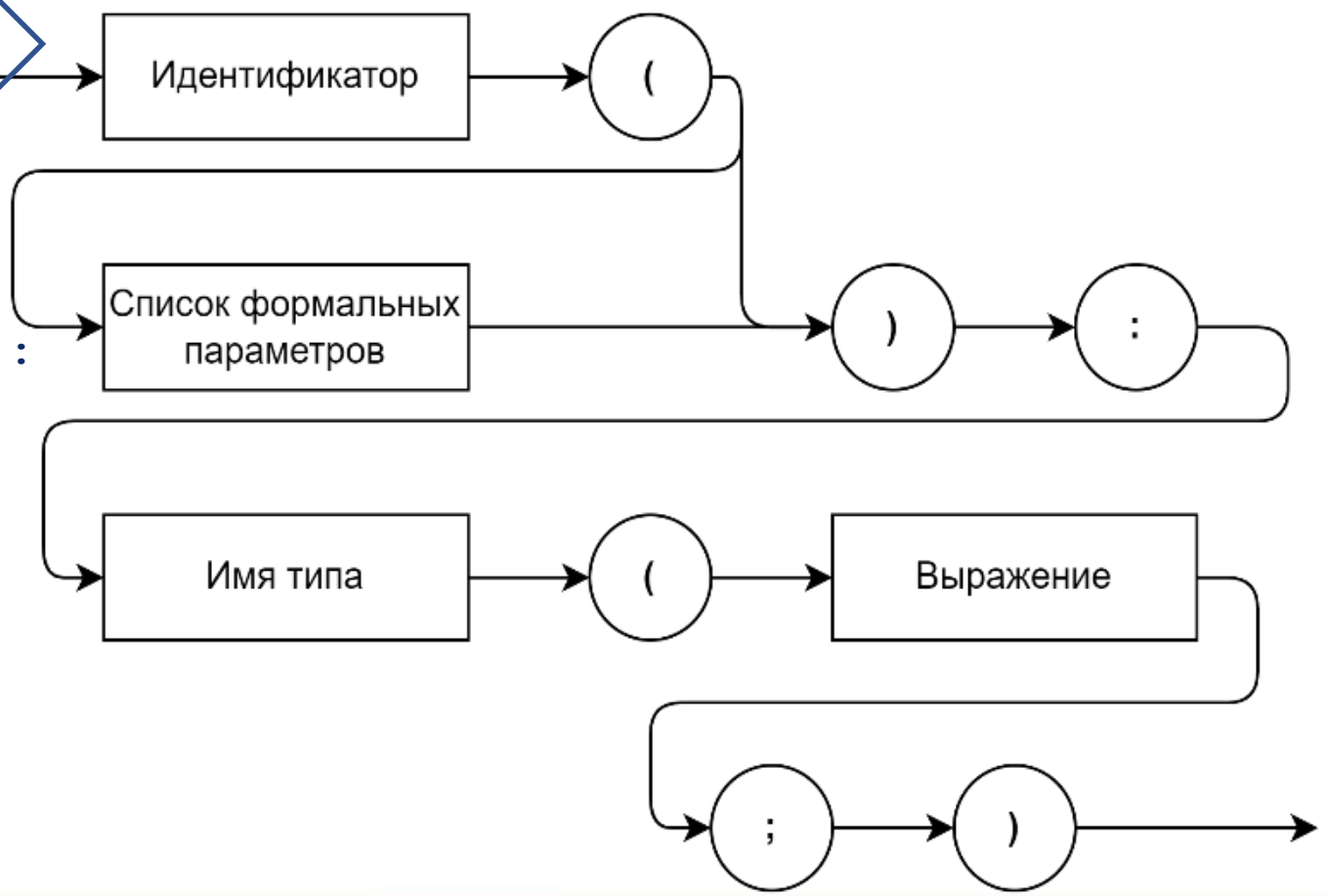
```
<правило обработки журнала событий>
        ::= <идентификатор>
        ( <идентификатор> : eventlog
        [ ; <параметр временного
        интервала> ] ) : eventlog ;
        ( FOREACH <идентификатор> ;
        <идентификатор> ;
        ( <выражение case-switch> ) )
```



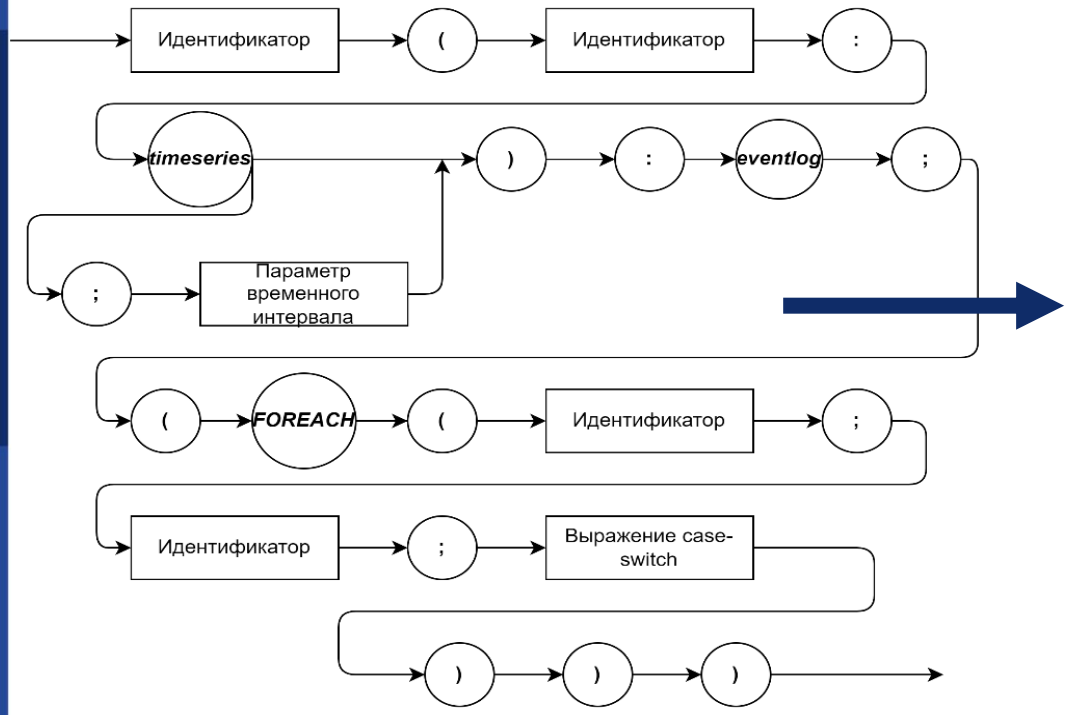
Формальный язык описания правил: пример

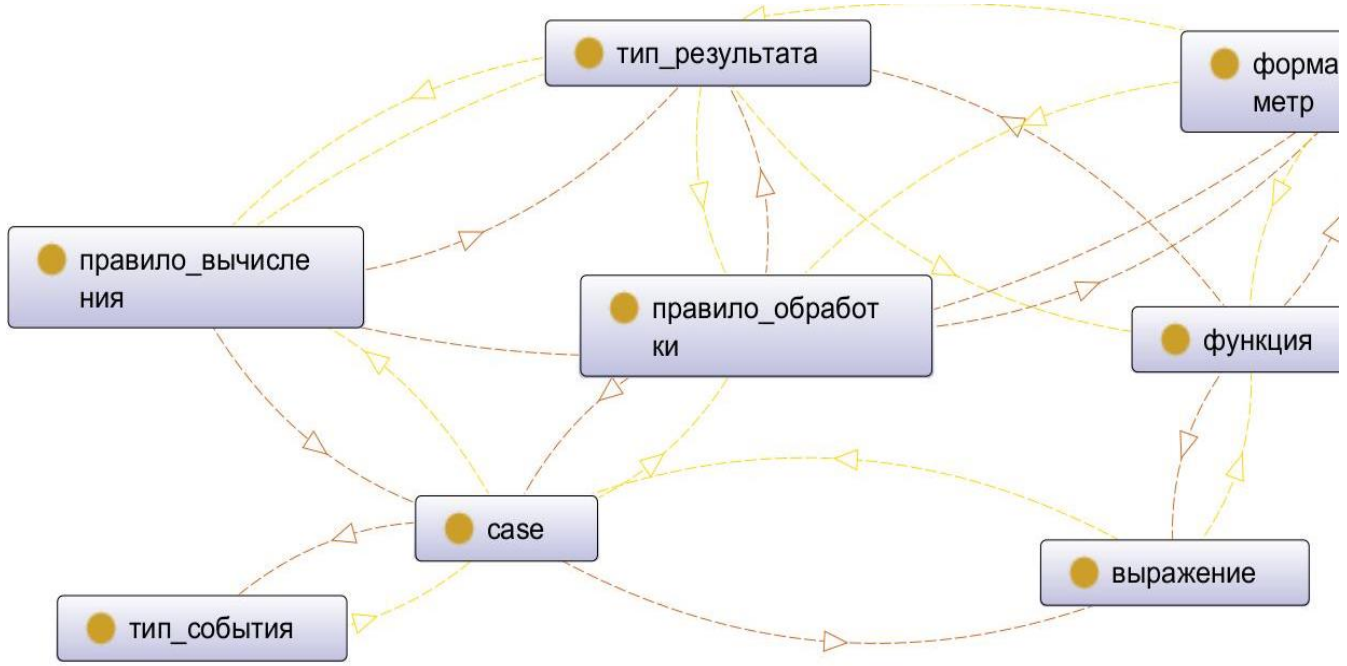
БНФ: → **Диаграмма Вирта**

```
<пользовательская функция> ::=
    <идентификатор>
    ( <список формальных параметров> ) :
    <имя типа> ;
    ( <выражение> )
```

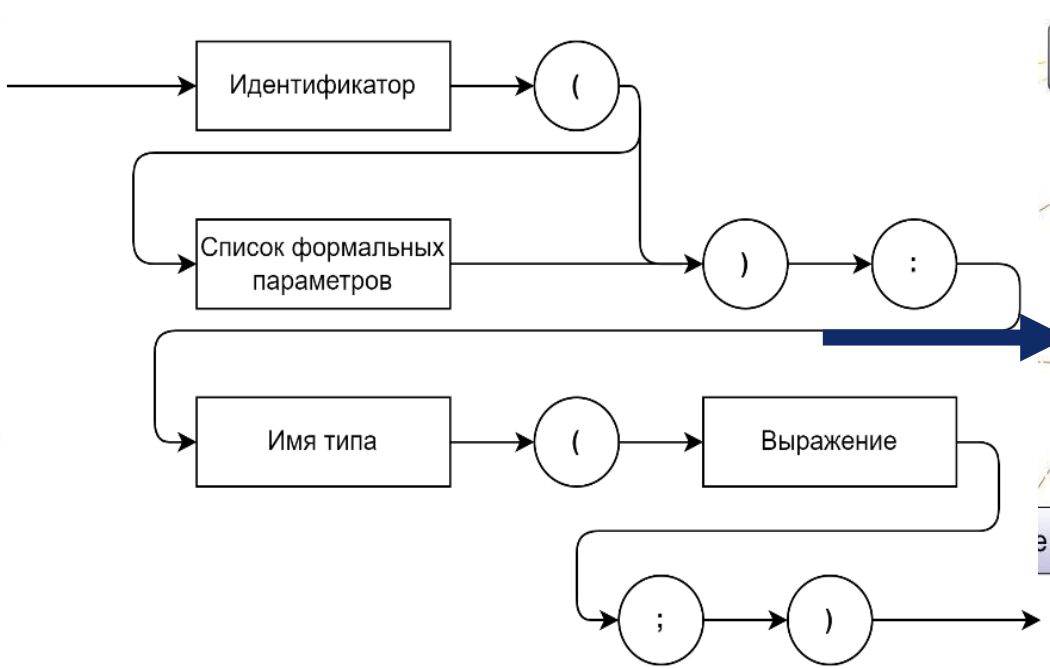


Онтология хранения правил и функций: пример



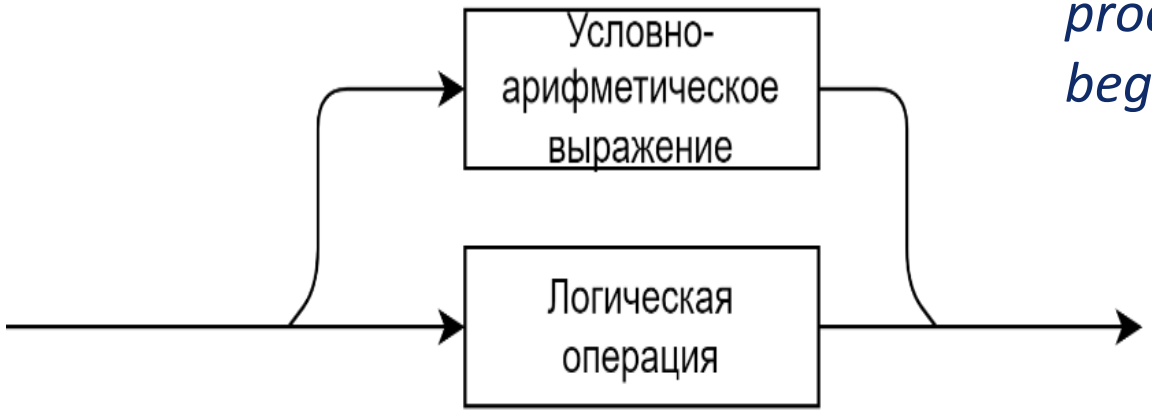


Онтология хранения правил и функций



Алгоритм синтаксического разбора: пример разбора выражения

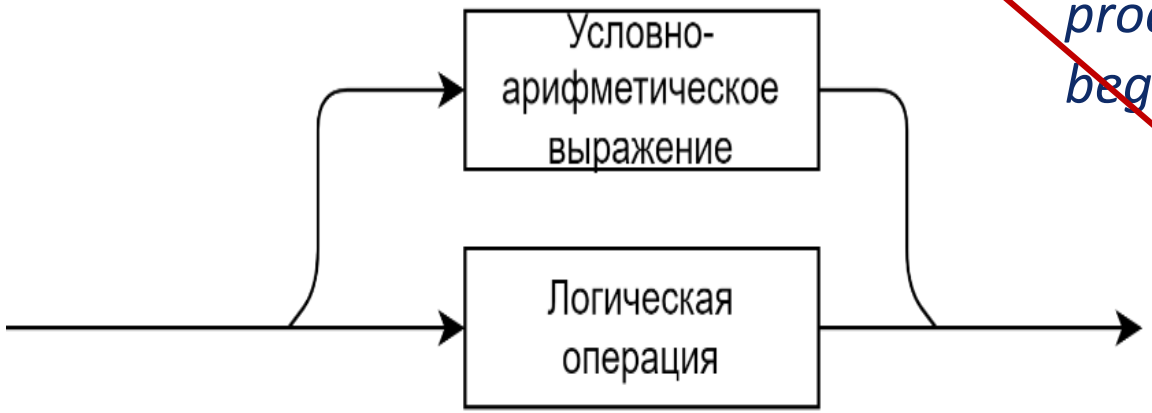
XOR (10 * 11 > 11 ; 20 + 11 < 10)




```
procedure ВЫРАЖЕНИЕ;
begin
    if Scansymbol in ЛОГИЧЕСКИЕ_ОПЕРАЦИИ then
    begin
            …
    end;
    else УСЛОВНО_АРИФМЕТИЧЕСКОЕ_ВЫРАЖЕНИЕ
end;
```

Алгоритм синтаксического разбора: пример разбора выражения

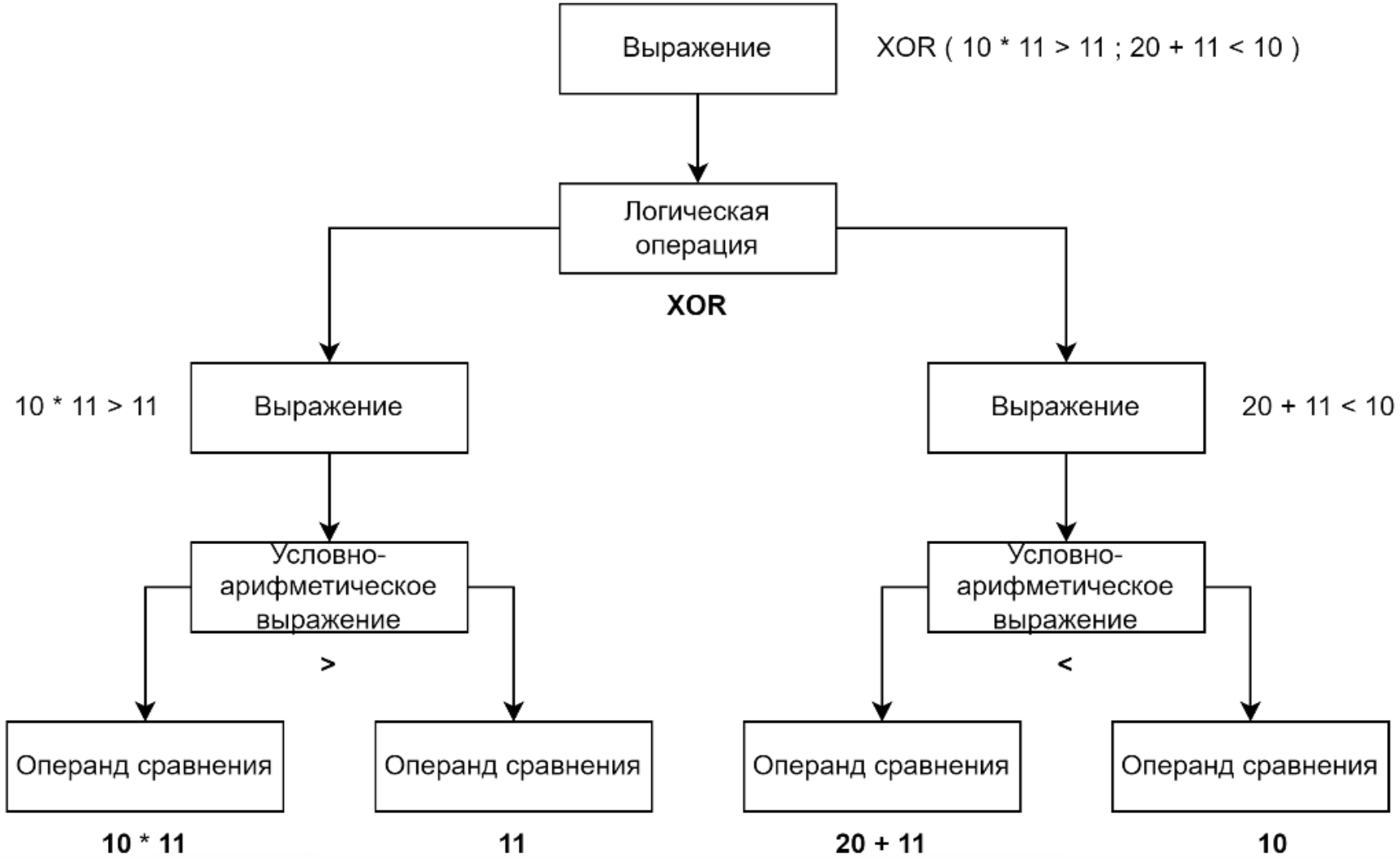
XOR (10 * 11 > 11 ; 20 + 11 < 10)




```
procedure ВЫРАЖЕНИЕ;
begin
    if Scansymbol in ЛОГИЧЕСКИЕ_ОПЕРАЦИИ then
    begin
        … /* Вызов процедуры разбора функции */
    end;
    else УСЛОВНО_АРИФМЕТИЧЕСКОЕ_ВЫРАЖЕНИЕ
    end;
```

Грамматика относится к классу *LL*(1) – по одному символу определяется нетерминальный символ грамматики и вызывается процедура разбора соответствующей конструкции в описании правила, сформированного пользователем с помощью конструктора. Для синтаксического разбора реализован алгоритм леворекурсивного спуска

Фрагмент дерева синтаксического разбора



Фрагмент дерева синтаксического разбора

В ходе синтаксического разбора для построенного правила формируется его представление в виде дерева, в которое включены все операции и их операнды. Сформированное представление правила включается в онтологию и может использоваться для генерации или обработки журналов событий с помощью разработанного интерпретатора.

Алгоритм интерпретации выражения

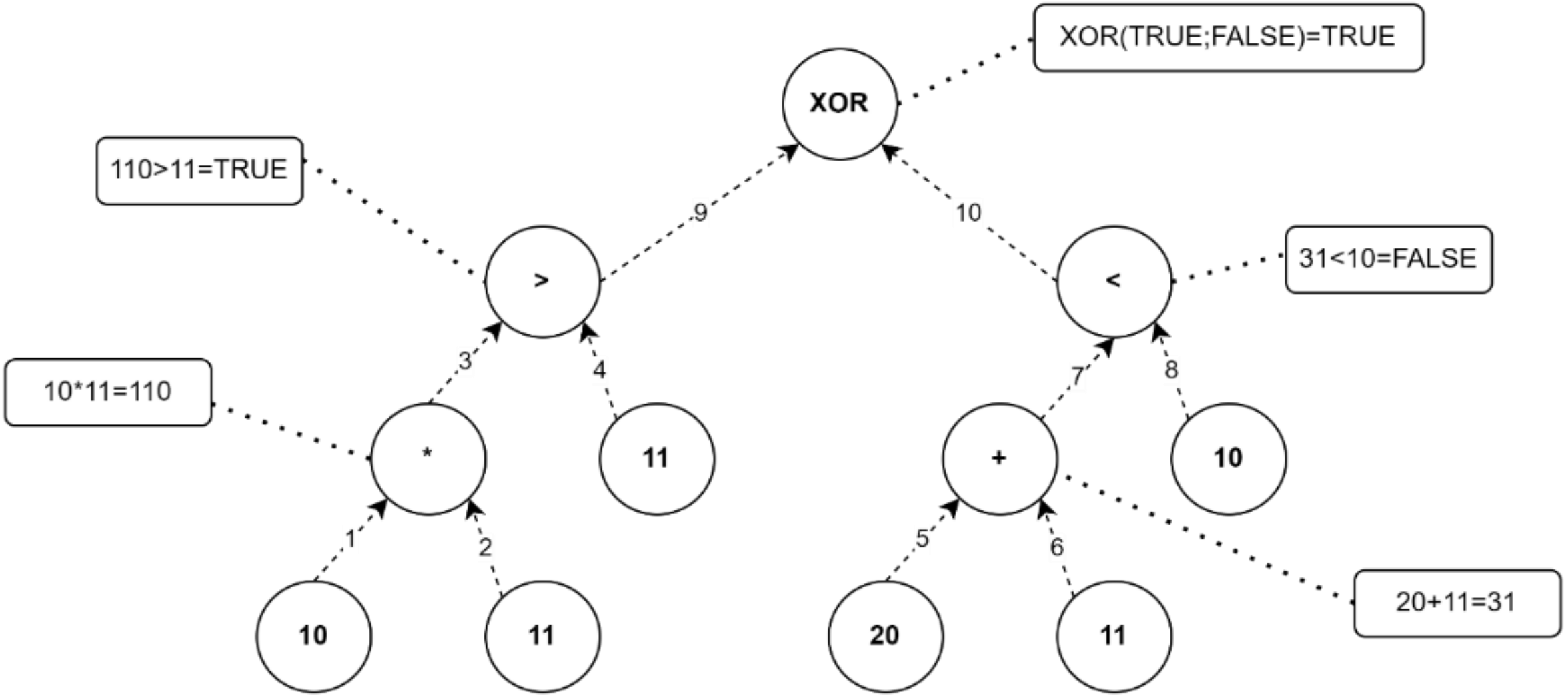
```
procedure CALCULATE;
begin
    if node in ОПЕРАТОРЫ then
    begin
        case node of
            '+': CALCULATE (node.Left) + CALCULATE (node.Right);
            '-': CALCULATE (node.Left) – CALCULATE (node.Right);
            '*': CALCULATE (node.Left) * CALCULATE (node.Right);
        …
end;
```

Алгоритм интерпретации выражения

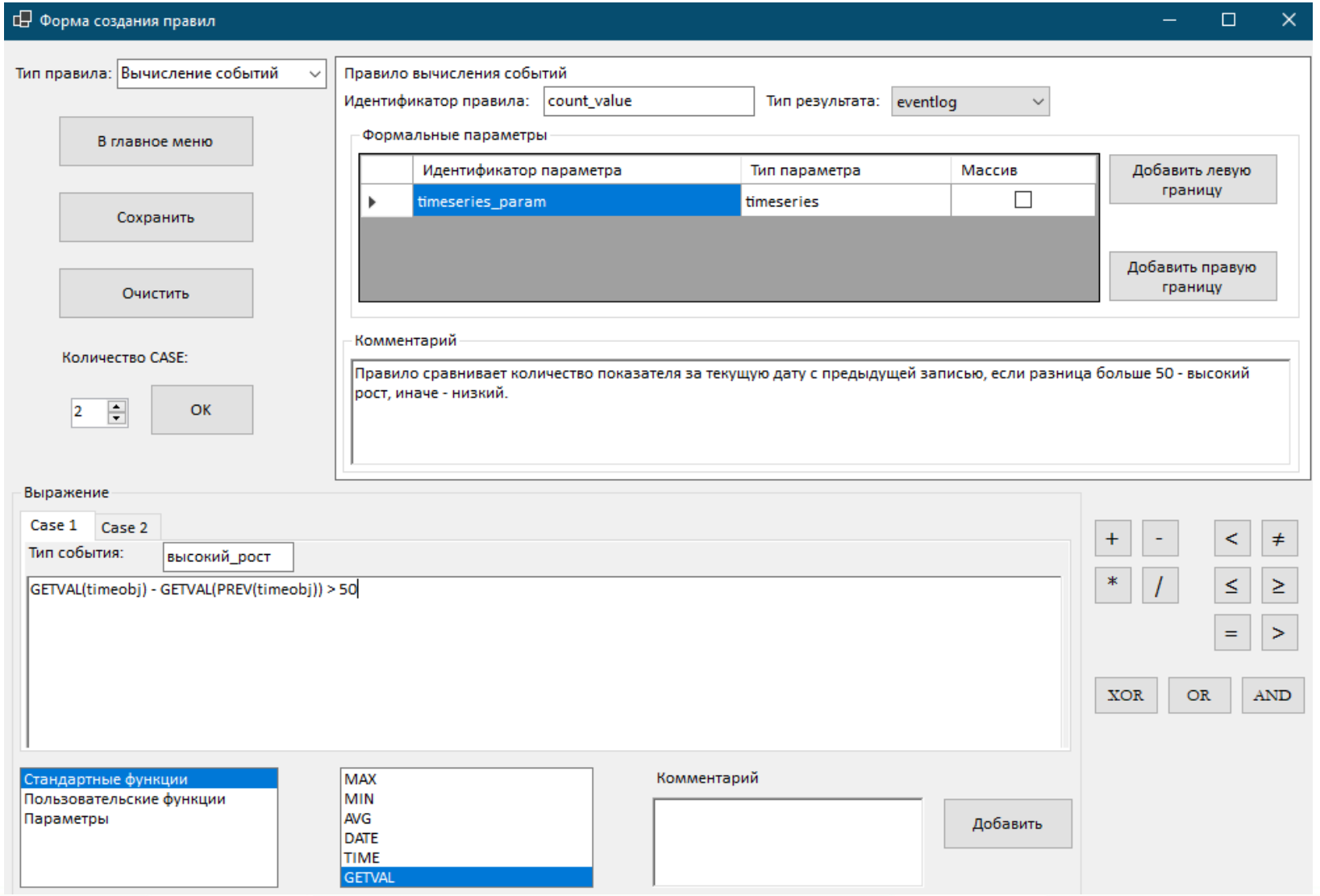
```
procedure CALCULATE;
begin
    if node in ОПЕРАТОРЫ then
        begin
            case node of
                '+': CALCULATE (node.Left) + CALCULATE (node.Right);
                '*': CALCULATE (node.Left) * CALCULATE (node.Right);
            ...
end;
```

Алгоритм интерпретации разработанных правил (функций) основан на рекурсивном обходе дерева, представляющего соответствующую функцию, построенную пользователем с помощью конструктора (сформированное правило включено в онтологию и может использоваться для формирования более сложных правил генерации журналов и их обработки)

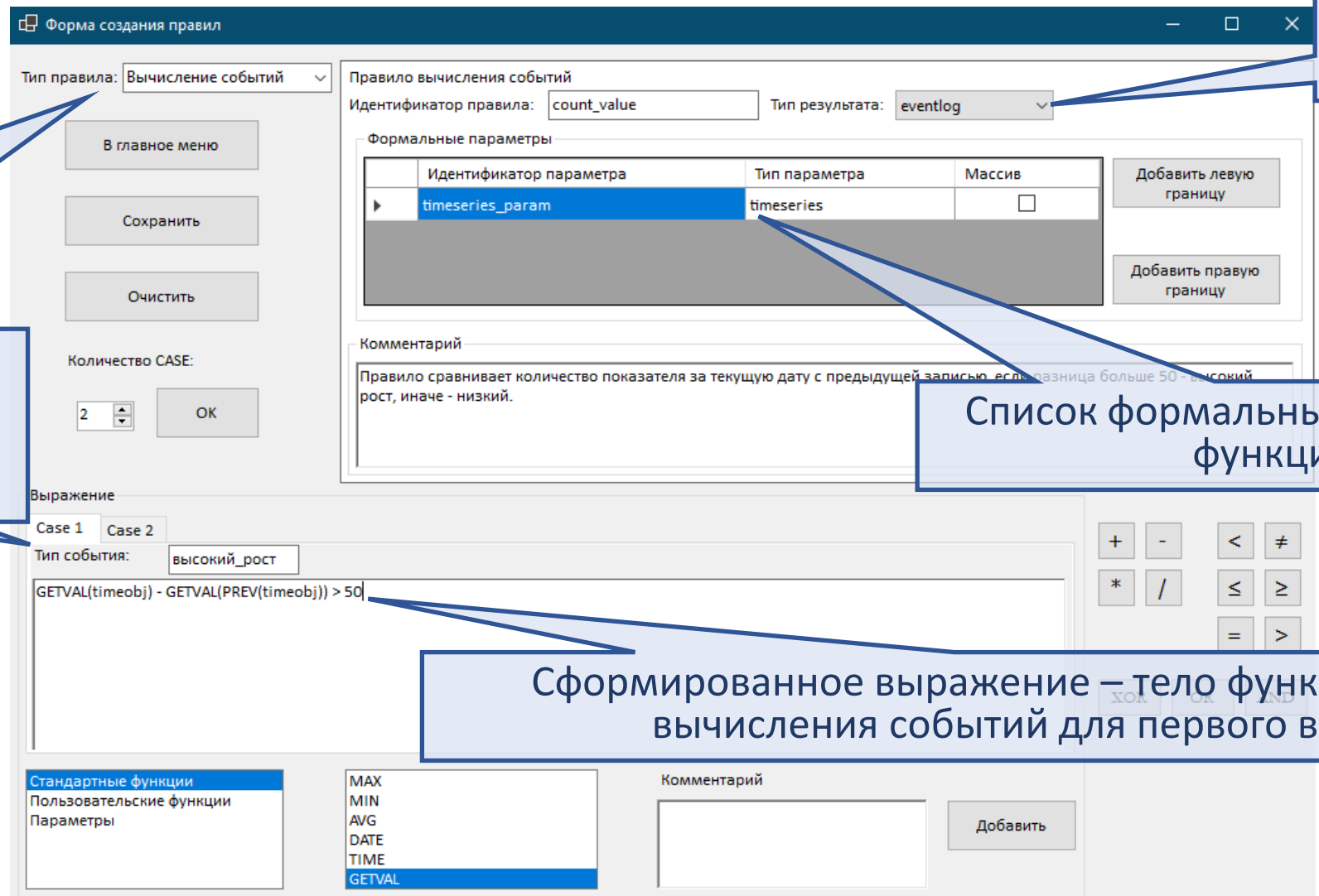
Пример интерпретации выражения на основе дерева выражения



Пример использования прототипа: конструктор правил



Пример использования прототипа: конструктор правил



Категория правила (функции)

Тип результата – журнал событий

Варианты вычисления в зависимости от условий

Список формальных параметров функции

Сформированное выражение – тело функции – правило вычисления событий для первого варианта

Пример использования прототипа: вычисление событий (интерпретация)

testcase_timeseries.csv – Блокнот

Файл Правка Формат Вид Сп

```
Timestamp,Value
2020-01-01 00:00:00,1230
2020-01-02 00:00:00,1270
2020-01-03 00:00:00,1328
2020-01-04 00:00:00,1373
2020-01-05 00:00:00,1447
2020-01-06 00:00:00,1511
2020-01-07 00:00:00,1553
2020-01-08 00:00:00,1615
2020-01-09 00:00:00,1679
2020-01-10 00:00:00,1723
2020-01-11 00:00:00,1783
2020-01-12 00:00:00,1843
2020-01-13 00:00:00,1910
2020-01-14 00:00:00,1979
2020-01-15 00:00:00,2050
2020-01-16 00:00:00,2105
2020-01-17 00:00:00,2156
2020-01-18 00:00:00,2215
```

Применение правила вычисления событий

Выберите правило вычисления событий

find_max_rule
правило_сравнения_минимумов
Value_compare_rule

Загрузить правило

Правило вычисления событий

Идентификатор правила: Value_compare_rule Тип результата: eventlog

Фактические параметры

Идентификатор параметра	Тип параметра	Массив	Значение
timeseries_param	timeseries	☐	Импортировано

Комментарий

Правило сравнивает значение текущей записи с предыдущей. Если различие больше 50 - высокий рост, меньше 50 - средний. Входной параметр: временной ряд.

Импорт временного ряда

Выполнить

Экспорт в XES

Временная метка	Значение параметра	Тип события
01.01.2020 0:00:00	1230	средний_рост
02.01.2020 0:00:00	1270	средний_рост
03.01.2020 0:00:00	1328	высокий_рост
04.01.2020 0:00:00	1373	средний_рост
05.01.2020 0:00:00	1447	высокий_рост
06.01.2020 0:00:00	1511	высокий_рост
07.01.2020 0:00:00	1553	средний_рост
08.01.2020 0:00:00	1615	высокий_рост
09.01.2020 0:00:00	1679	высокий_рост

Пример использования прототипа – интерпретатора правил: вычисление событий (генерация журнала)

testcase_timeseries.csv – Блокнот

Файл Правка Формат Вид Сп

```
Timestamp,Value
2020-01-01 00:00:00,1230
2020-01-02 00:00:00,1270
2020-01-03 00:00:00,1328
2020-01-04 00:00:00,1373
2020-01-05 00:00:00,1447
2020-01-06 00:00:00,1511
2020-01-07 00:00:00,1553
2020-01-08 00:00:00,1615
2020-01-09 00:00:00,1679
2020-01-10 00:00:00,1723
2020-01-11 00:00:00,1783
2020-01-12 00:00:00,1843
2020-01-13 00:00:00,1910
2020-01-14 00:00:00,1979
2020-01-15 00:00:00,2060
2020-01-16 00:00:00,2105
2020-01-17 00:00:00,2156
2020-01-18 00:00:00,2215
```

Исходные данные – фактический параметр функции

Сформированный журнал событий

Применение правила вычисления событий

Выберите правило вычисления событий

find_max_rule
правило_сравнения_минимумов
Value_compare_rule

Загрузить правило

Правило вычисления событий

Идентификатор правила: Value_compare_rule Тип результата: eventlog

Фактические параметры

Идентификатор параметра	Тип параметра	Массив	Значение
timeseries_param	timeseries	☐	Импортировано

Комментарий

Правило сравнивает значение текущей записи с предыдущей. Если различие больше 50 - высокий рост, меньше 50 - средний. Входной параметр: временной ряд.

Импорт временного ряда

Выполнить

Экспорт в XES

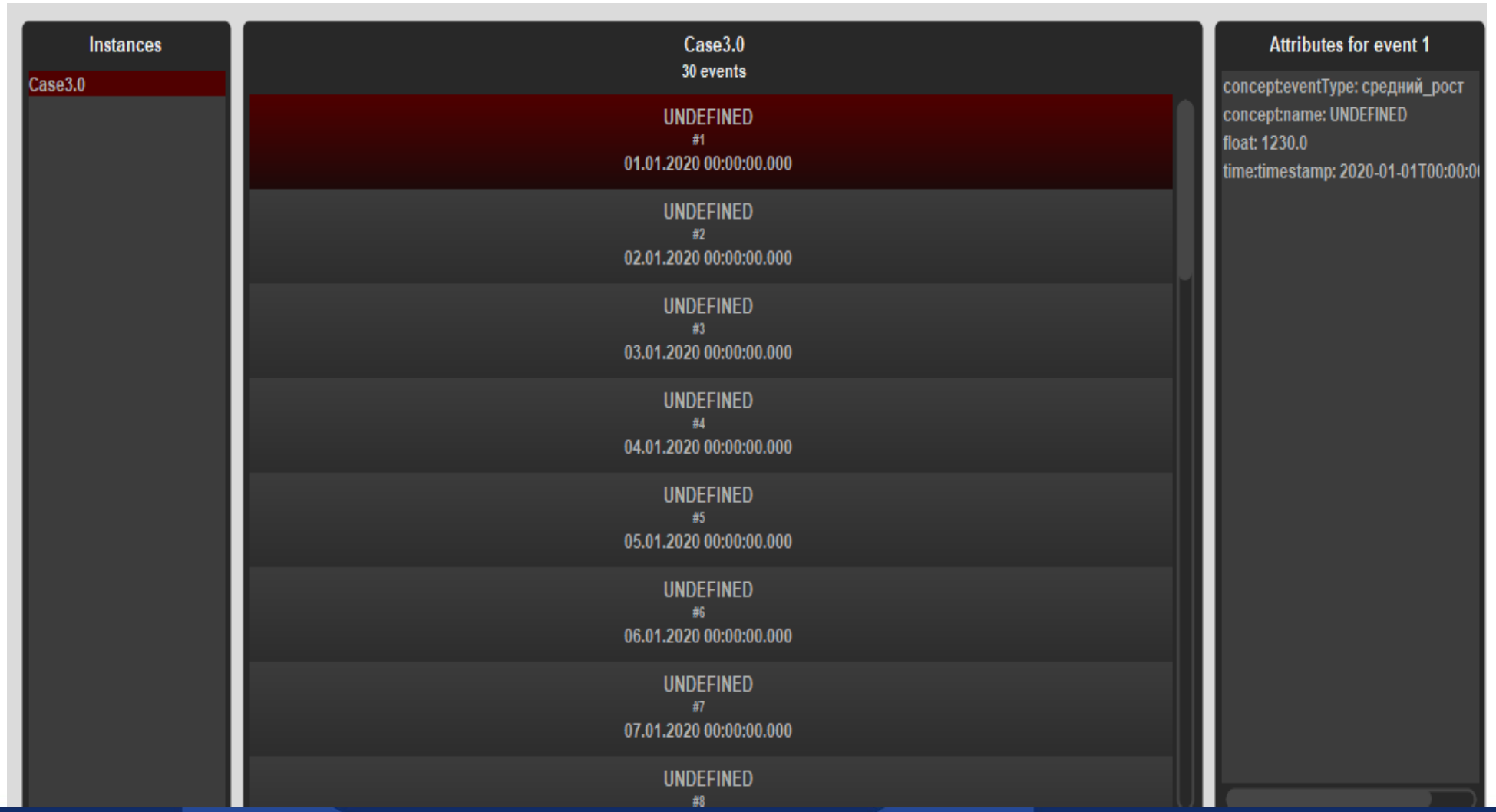
Временная метка	Значение параметра	Тип события
01.01.2020 0:00:00	1230	средний_рост
02.01.2020 0:00:00	1270	средний_рост
03.01.2020 0:00:00	1328	высокий_рост
04.01.2020 0:00:00	1373	средний_рост
05.01.2020 0:00:00	1447	высокий_рост
06.01.2020 0:00:00	1511	высокий_рост
07.01.2020 0:00:00	1553	средний_рост
08.01.2020 0:00:00	1615	высокий_рост
09.01.2020 0:00:00	1679	высокий_рост

Пример использования прототипа: генерация журнала событий (XES)

```xml
<?xml version="1.0" encoding="utf-8"?>
<log xes.version="1.0" xes.features="nested-attributes" xmlns:x="http://www.xes-standard.org/">
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext" />
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext" />
  <global scope="trace">
    <string key="concept:name" value="__INVALID__" />
  </global>
  <global scope="event">
    <string key="concept:name" value="__INVALID__" />
    <string key="concept:eventType" value="__INVALID__" />
  </global>
  <string key="source" value="EventInterpetor" />
  <trace>
    <string key="concept:name" value="Case3.0" />
    <event>
      <string key="concept:name" value="UNDEFINED" />
      <date key="time:timestamp" value="2020-01-01T00:00:00.000" />
      <float key="float" value="1230" />
      <string key="concept:eventType" value="средний_рост" />
    </event>
    <event>
      <string key="concept:name" value="UNDEFINED" />
      <date key="time:timestamp" value="2020-01-02T00:00:00.000" />
      <float key="float" value="1270" />
      <string key="concept:eventType" value="средний_рост" />
    </event>
    <event>
      <string key="concept:name" value="UNDEFINED" />
      <date key="time:timestamp" value="2020-01-03T00:00:00.000" />
      <float key="float" value="1328" />
      <string key="concept:eventType" value="высокий_рост" />
    </event>
    <event>
```

Фрагмент сформированного журнала событий в формате XES

Пример использования прототипа: загрузка сгенерированного журнала в ProM



Заключение

Создана среда разработки, которая позволяет разрабатывать алгоритмы анализа событийных рядов с помощью конструктора пользователям, не владеющим навыками программирования.

Практическая значимость – разработанные средства позволяют:

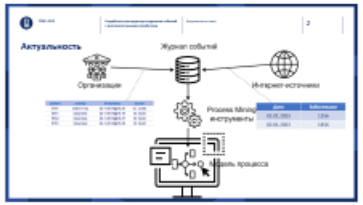
- Создавать функции для выявления закономерностей в цепочных событиях без привлечения программистов (уровень требований к пользователю – умение работать с построителями выражений в MS Office).
- Обрабатывать дополнительную информацию (дополнительные атрибуты событий) при рассмотрении количественных и числовых показателей с помощью конструктора по принципу «low-code».

Содержание

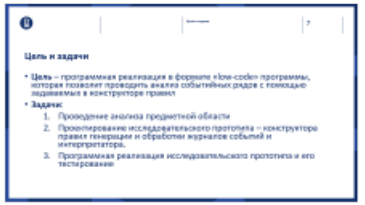

НИУ ВШЭ – Пермь, ПГНИУ

РАЗРАБОТКА КОНСТРУКТОРА ЖУРНАЛОВ СОБЫТИЙ С ДОПОЛНИТЕЛЬНЫМИ АТРИБУТАМИ

ТРИС, 2023

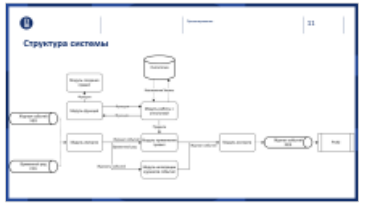

Актуальность


Объект и предмет


Цель и задачи


Функциональные требования


Функциональные требования – диаграмма прецедентов

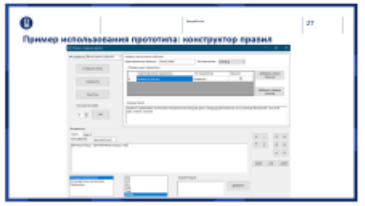

Структура системы


Представление правил генерации и обработки журналов событий


Онтология хранения правил и функций: пример


Алгоритм синтаксического разбора: пример разбора выражения


Алгоритм интерпретации выражения


Пример использования прототипа: конструктор правил


Заключение

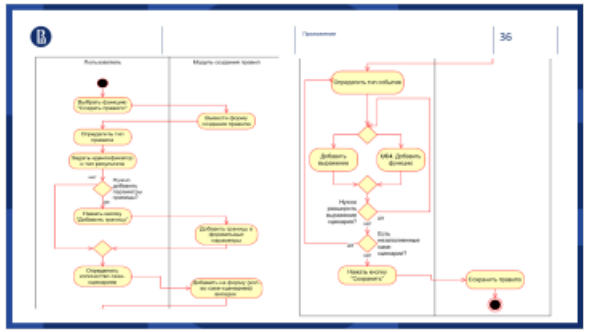
Создана среда разработки, которая позволяет разрабатывать алгоритмы анализа событийных рядов с помощью конструктора пользователем, не владеющим навыками программирования.

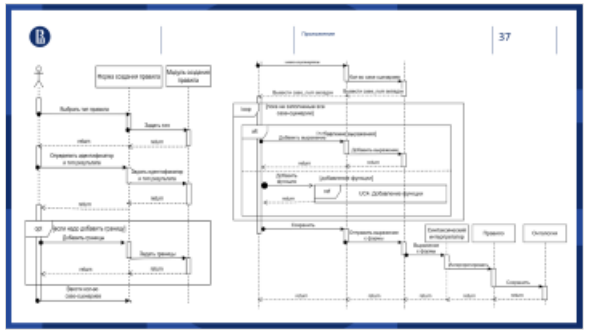
Практическая значимость – разработанные средства позволяют:

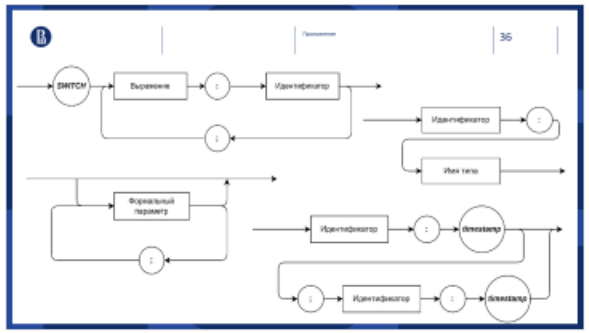
- Создавать функции для выявления закономерностей в целевых событиях без привлечения программистов (уровень требований к пользователю – умение работать с построителями выражений в MS Office).
- Обрабатывать дополнительную информацию (дополнительные атрибуты событий) при рассмотрении количественных и числовых показателей с помощью конструктора по принципу «low-code».

Приложения

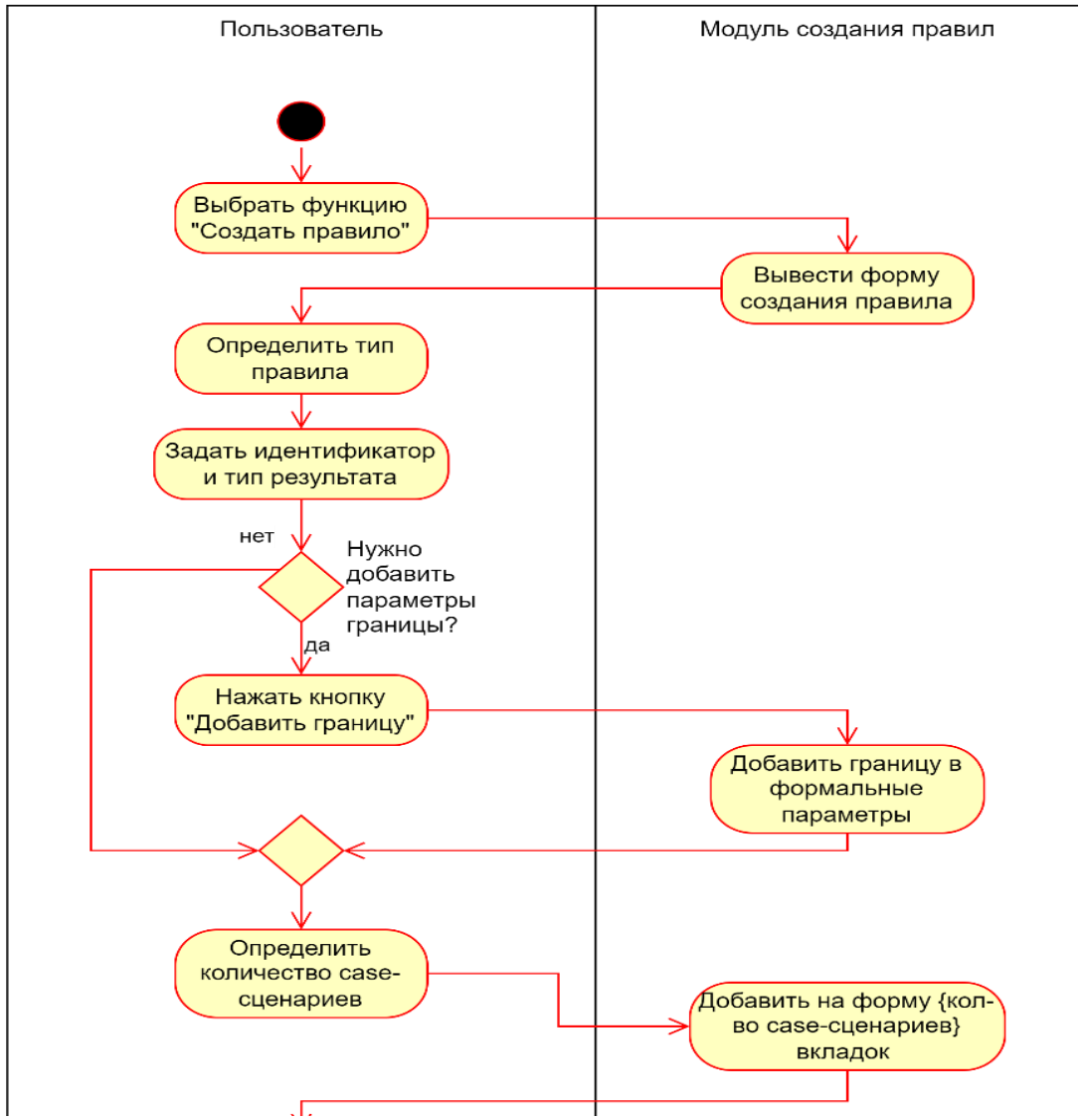
Приложения

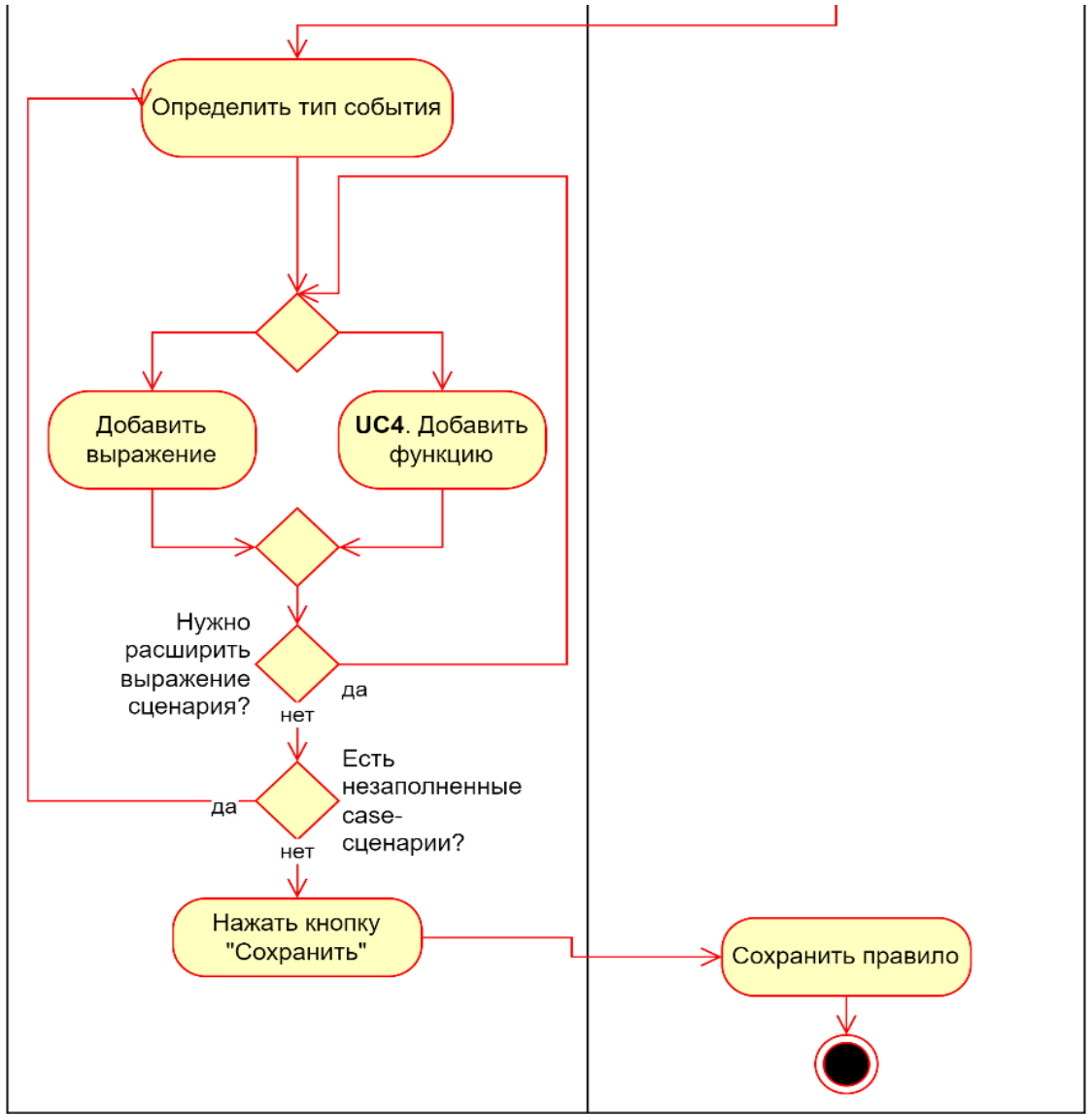


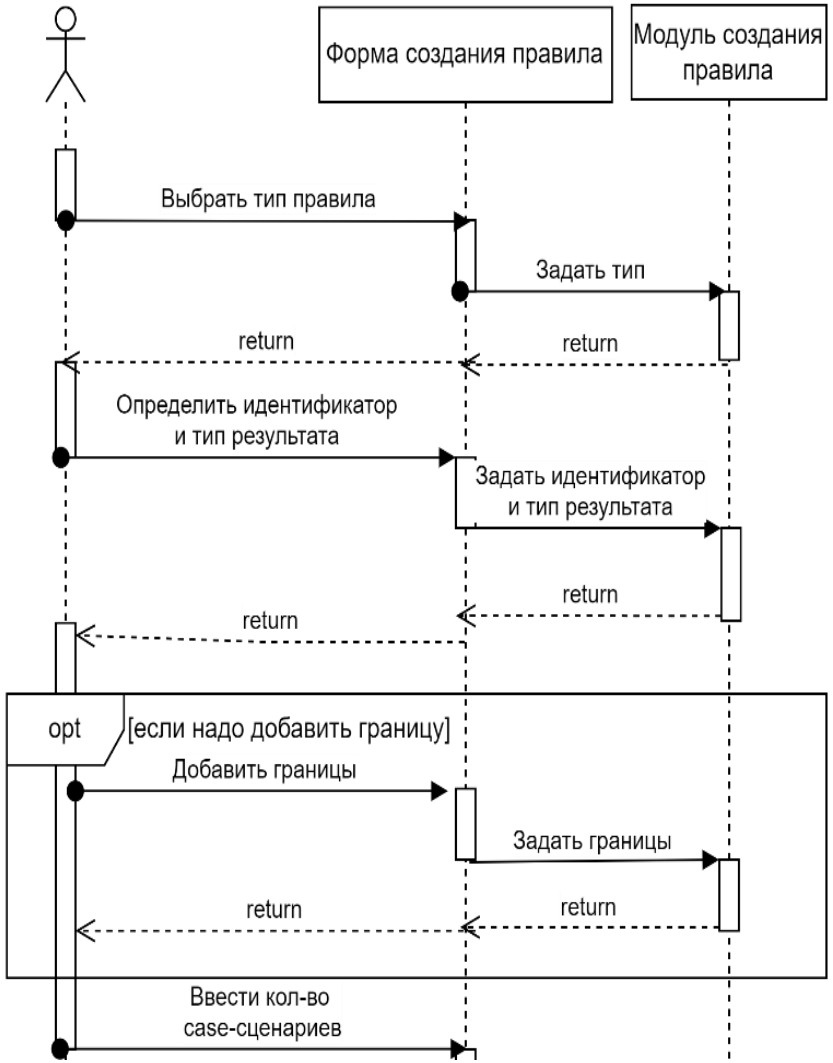


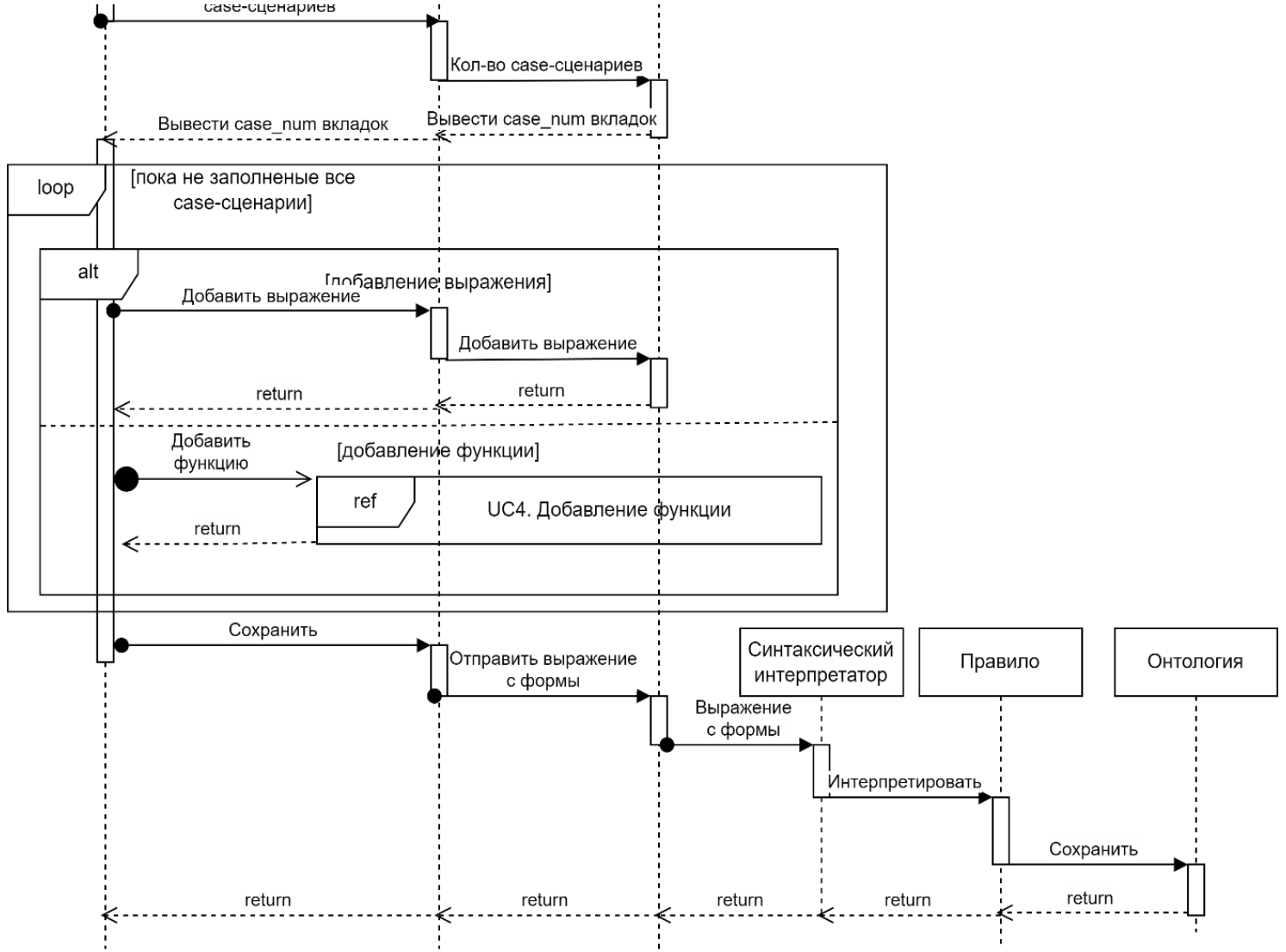


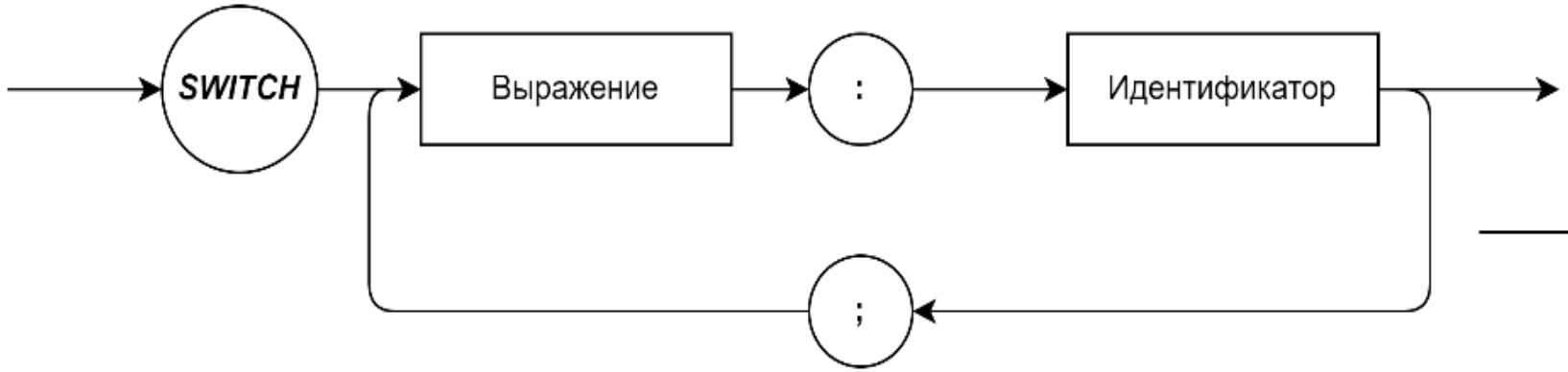
Содержание


Пользователь
Модуль создания правил
Выбрать функцию "Создать правило"
Вывести форму создания правила
Определить тип правила
Задать идентификатор и тип результата
нет
Нужно добавить параметры границы?
да
Нажать кнопку "Добавить границу"
Добавить границу в формальные параметры
Определить количество case-сценариев
Добавить на форму {кол-во case-сценариев} вкладок

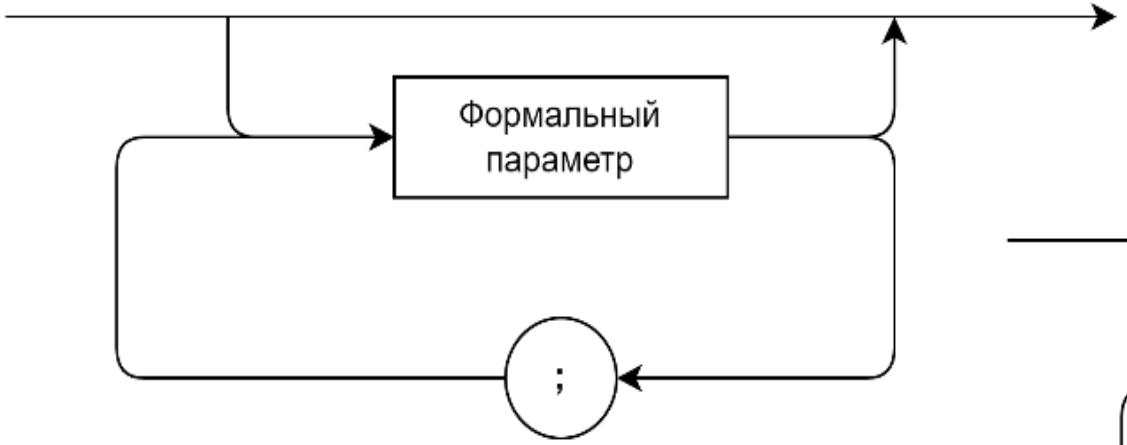

Определить тип события
Добавить выражение
UC4. Добавить функцию
Нужно расширить выражение сценария?
да
нет
Есть незаполненные case-сценарии?
да
нет
Нажать кнопку "Сохранить"
Сохранить правило

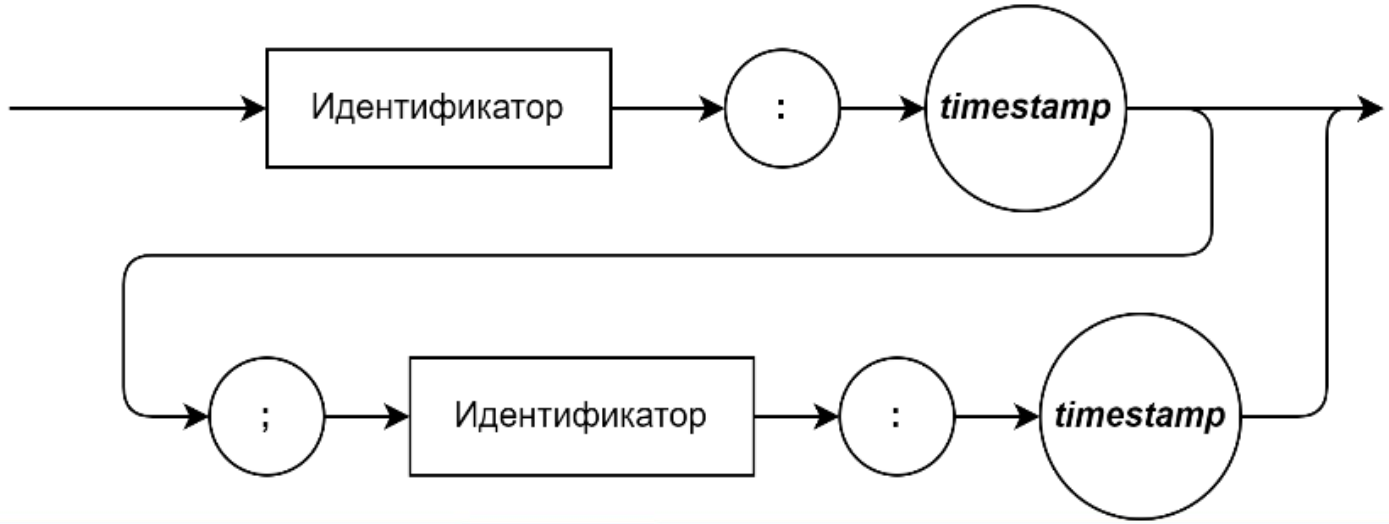

Форма создания правила
Модуль создания правила
Выбрать тип правила
Задать тип
return
return
Определить идентификатор и тип результата
Задать идентификатор и тип результата
return
return
opt
[если надо добавить границу]
Добавить границы
Задать границы
return
return
Ввести кол-во case-сценариев

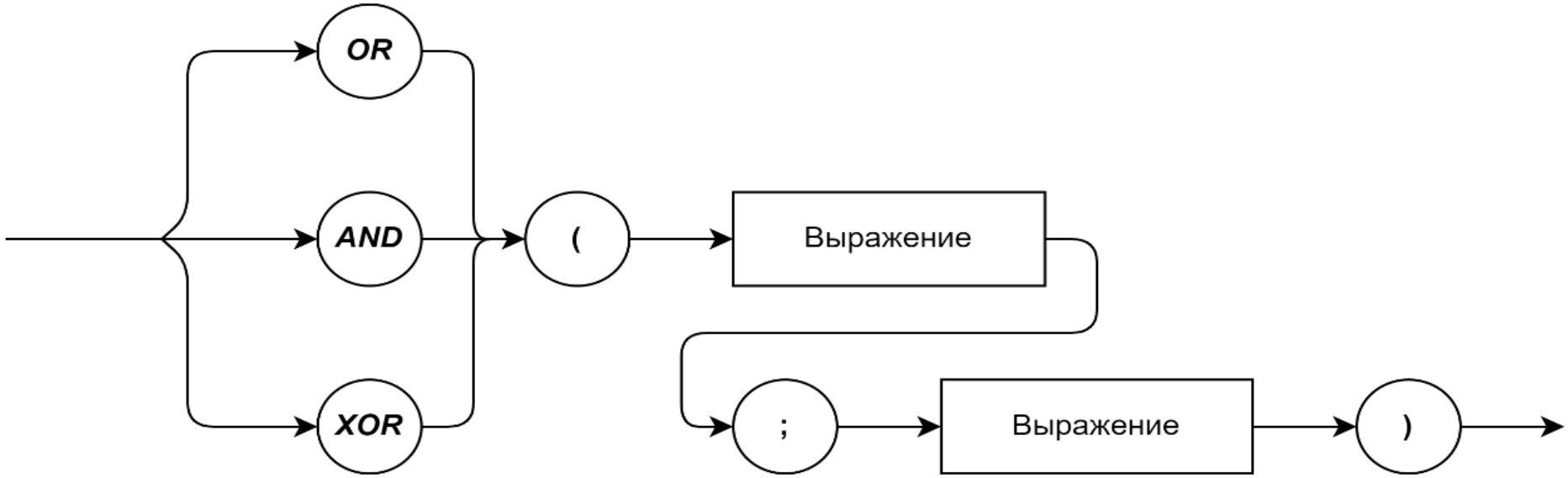

case-сценариев
Кол-во case-сценариев
Вывести case_num вкладок
Вывести case_num вкладок
loop
[пока не заполненые все case-сценарии]
alt
[добавление выражения]
Добавить выражение
Добавить выражение
return
return
Добавить функцию
[добавление функции]
ref
UC4. Добавление функции
return
Сохранить
Отправить выражение с формы
Синтаксический интерпретатор
Правило
Онтология
Выражение с формы
Интерпретировать
Сохранить
return
return
return
return
return

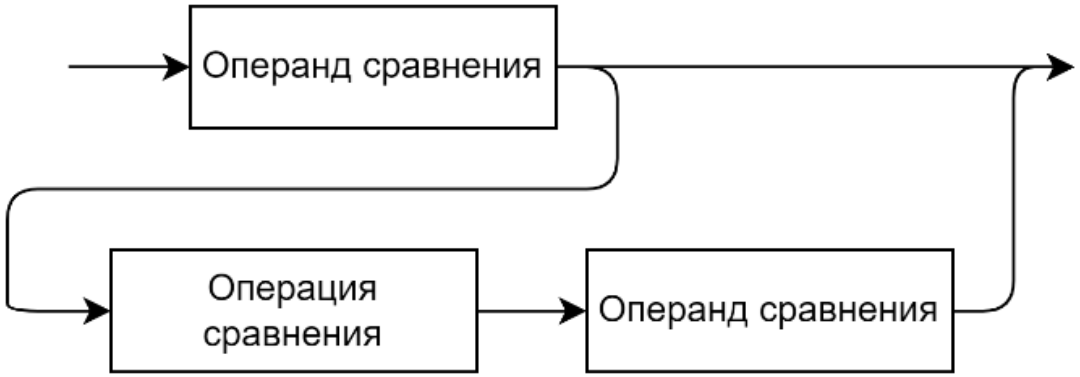

SWITCH
Выражение
:
Идентификатор
;


Идентификатор
:
Имя типа

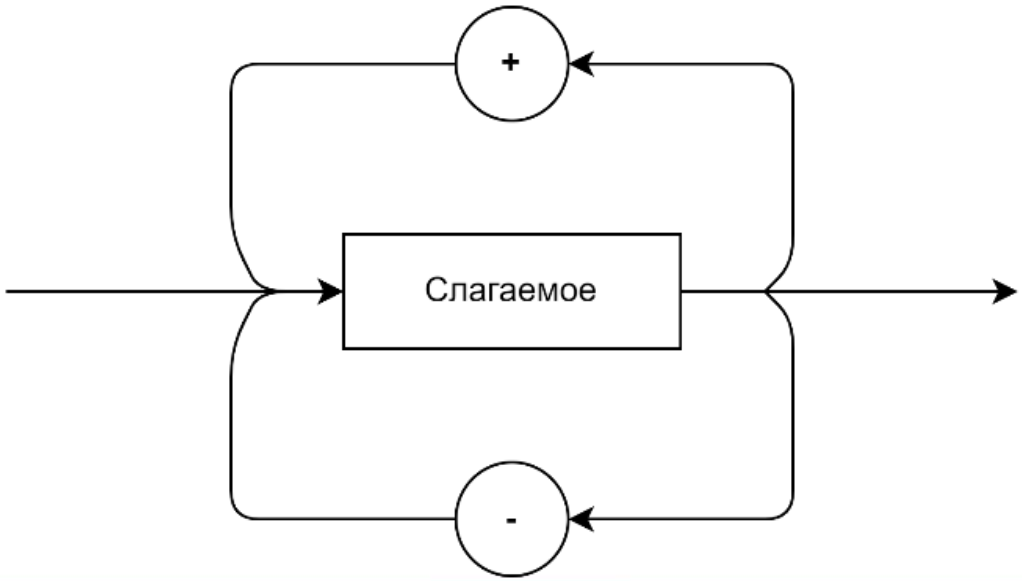

Формальный параметр
;

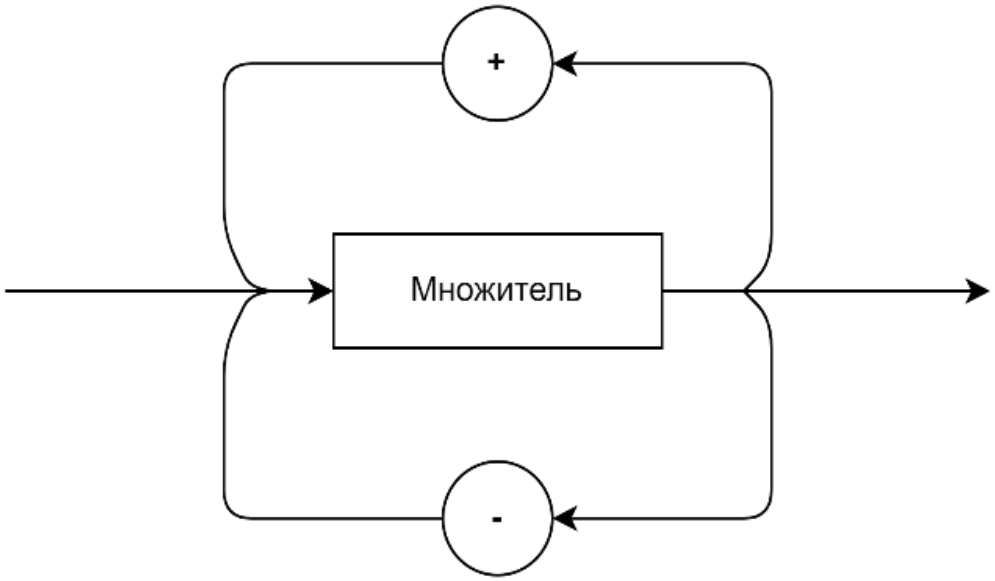

Идентификатор
:
timestamp
;
Идентификатор
:
timestamp

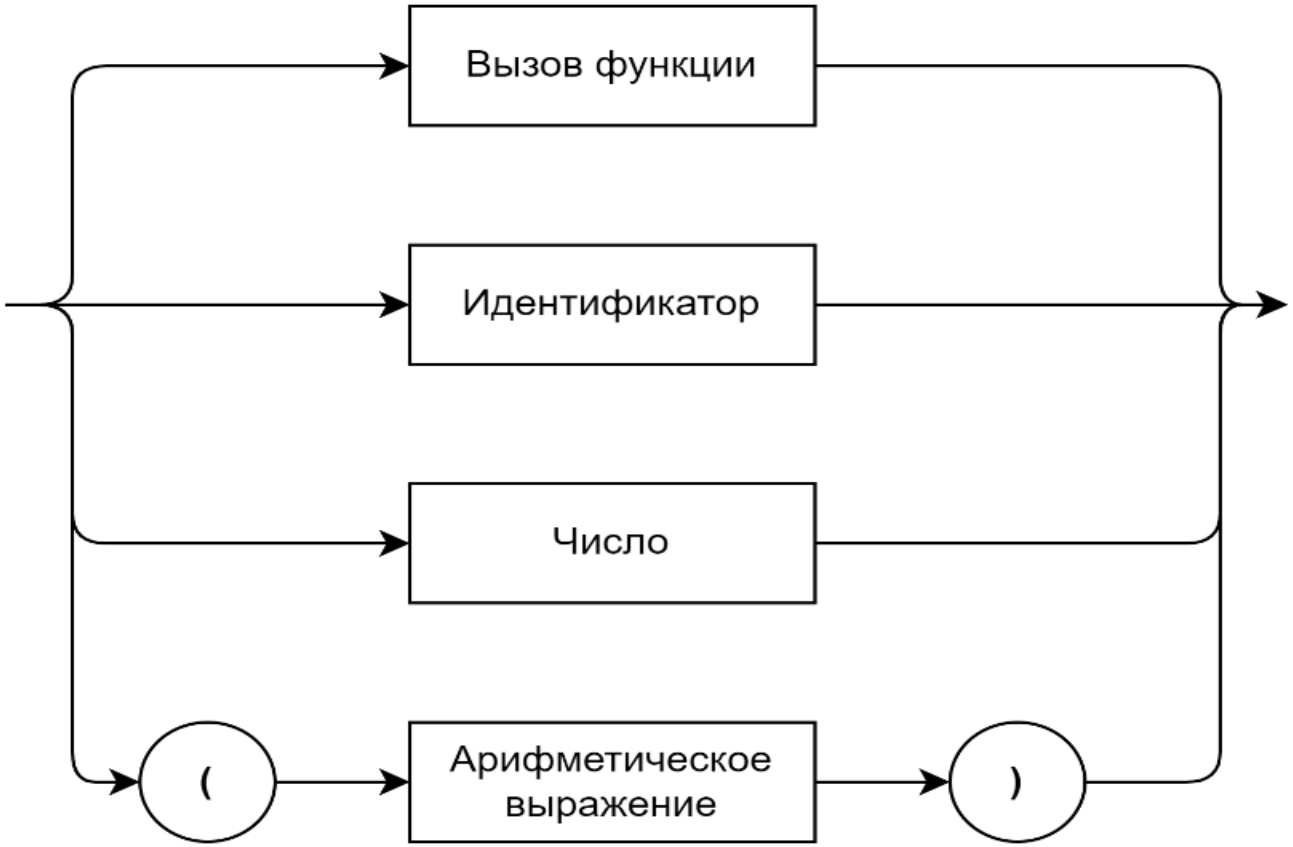

OR
AND
XOR
(
Выражение
;
Выражение
)


Операнд сравнения
Операция сравнения
Операнд сравнения


Арифметическое выражение
(
)


+
Слагаемое
-


+
Множитель
-


Вызов функции
Идентификатор
Число
(
Арифметическое выражение
)