Прототип базы активных знаний

Зимняя школа 2019

Руководитель: Перепёлкин Владислав Александрович

Выполнили: Артюхов Алексей Андреевич
4 курс ФИТ НГУ

Парфёнов Денис Романович
2 курс ФИТ НГУ

1 февраля 2019

План доклада

1. Задача
2. Идея решения
3. Реализация
4. Тестирование
5. Заключение

Постановка задачи

В текущий момент времени существует проблема хранения знаний. Они должны храниться в библиотеках, таким образом, чтобы их можно было максимально эффективно использовать.

Таким решением является База Активных Знаний.

Идея решения

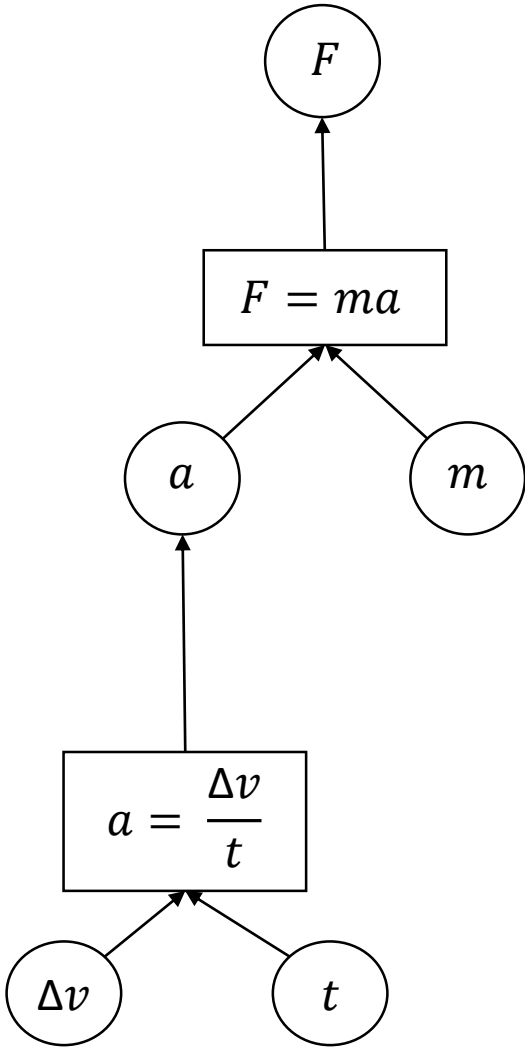
БАЗ связана с некоторой предметной областью через её формализацию в виде множества переменных (величин предметной области) и подпрограмм, которые является операциями над этими переменными, связывают их, вычисляя одни величины из других.

Идея решения

Пользователь указывает какие величины он должен получить и какие ему известны, а БАЗа пытается подобрать подпрограмму или набор подпрограмм, с помощью которых возможно вычислить выходные значения из входных и генерирует план этих вычислений.

Дано: $\Delta v, t, m$. Вычислить: F.

Результат: $F = m \times \frac{\Delta v}{t}$



Реализация

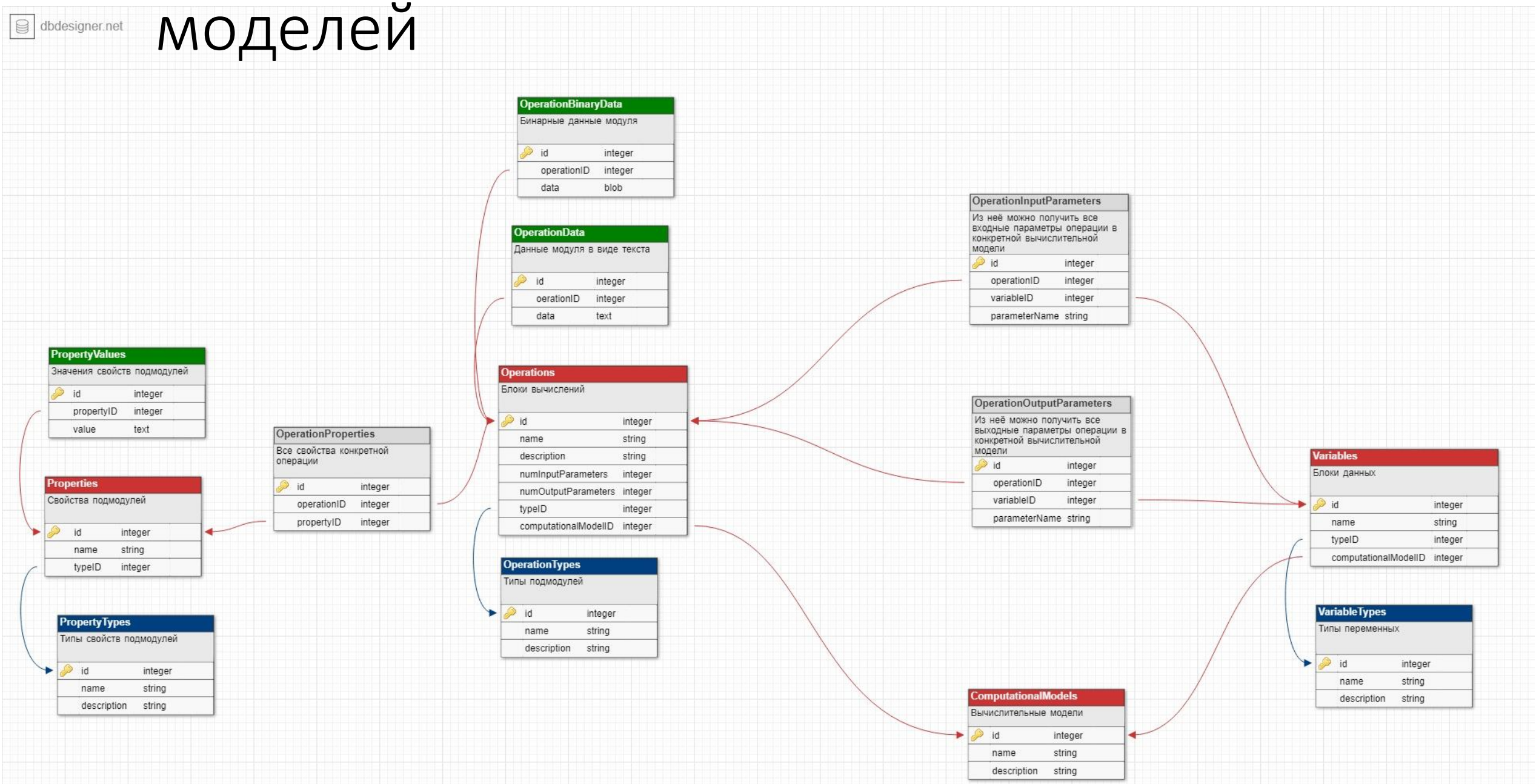
БАЗа состоит из трёх основных компонент:

1. Хранилище вычислительных моделей
2. Модуль вывода плана вычислений
3. Модуль реализации плана вычислений

Мы предлагаем хранить для каждой предметной области граф, который свяжет между собой различные модули и величины, вычисляемые ими. Пользователь добавляет в систему модули, и они автоматически связываются по существующим переменным с графом.

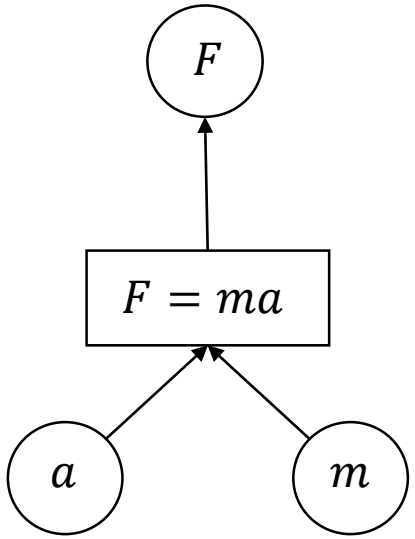
Для реализации был выбран язык C++. Для работы с графами была использована Boost Graph Library. Для работы с базами данных применялась QtSql.

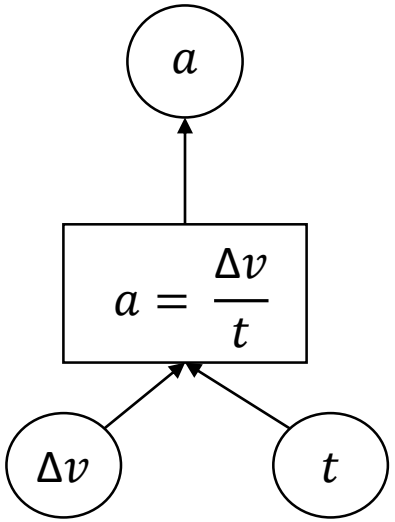
Реализация хранилища вычислительных моделей

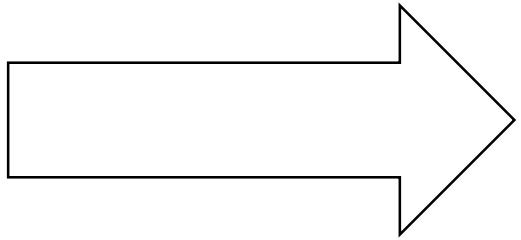


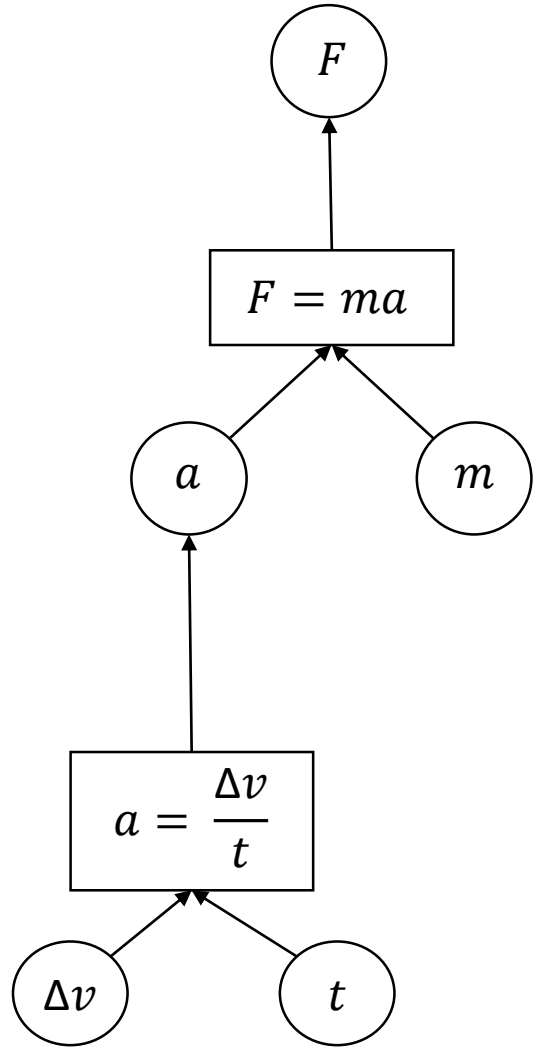
Реализация (База данных)

При первом запуске системы хранения, осуществляется загрузка и запись в базу данных вычислительных моделей, операций и переменных из соответствующих директорий. После чего, появляется возможность запрашивать у системы хранения вычислительные модели собранные в графы.


F
$F = ma$
a
m


a
$a = \frac{\Delta v}{t}$
Δv
t




F
$F = ma$
a
m
$a = \frac{\Delta v}{t}$
Δv
t

Реализация (Модуль вывода плана вычислений)

На вход поступает граф и два списка вершин (известные величины и которые необходимо вычислить).

Выходом является последовательный план вычислений.

Данный модуль можно разбить на два этапа:

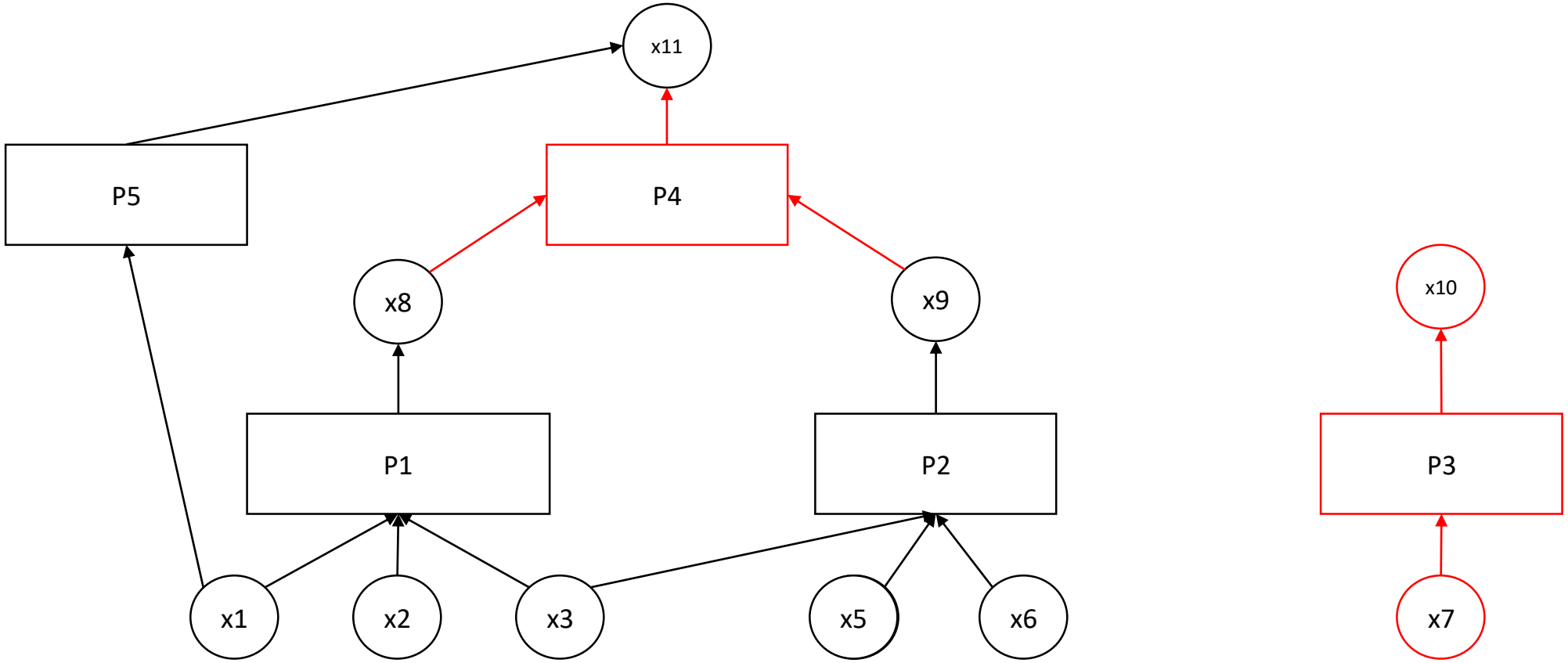
1. Построение дерева вычислений
2. Получение плана вычислений с помощью обхода дерева

Реализация (Построение дерева вычислений)

Алгоритм состоит из 2 частей:

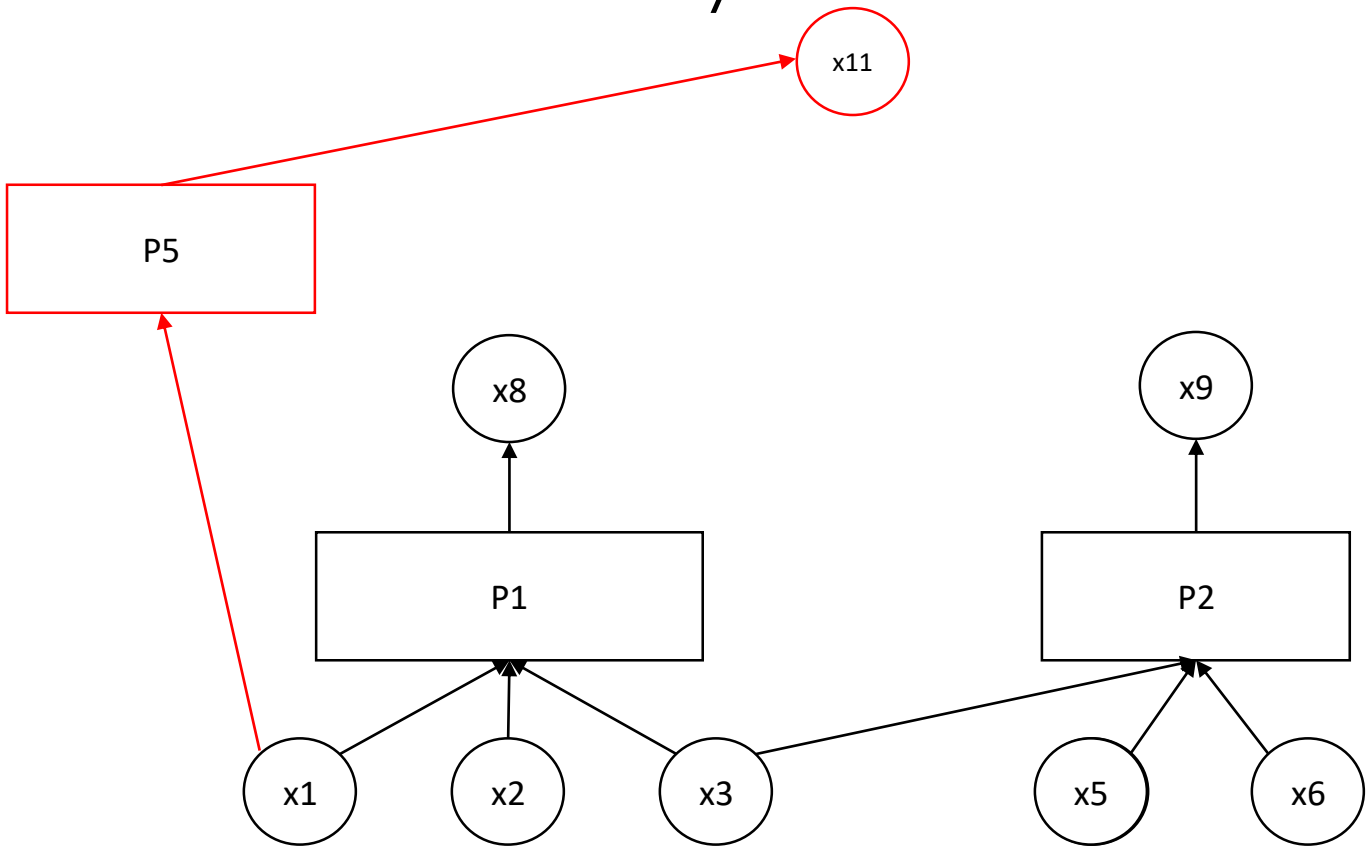
1) Восходящая часть: известные вершины помечаются как «вычислимые», далее в список «вычислимых» добавляются вершины, которые можно вычислить с помощью модулей из «вычислимых». Так продолжается либо пока все необходимые для получения вершины будут помечены как «вычислимые», либо пока не останется необработанных смежных с «вычислимыми» вершин (это будет означать отсутствие решения)

std::vector<T> in = {x1,x2,x3,x5,x6}; std::vector<T> req = {x8, x9}

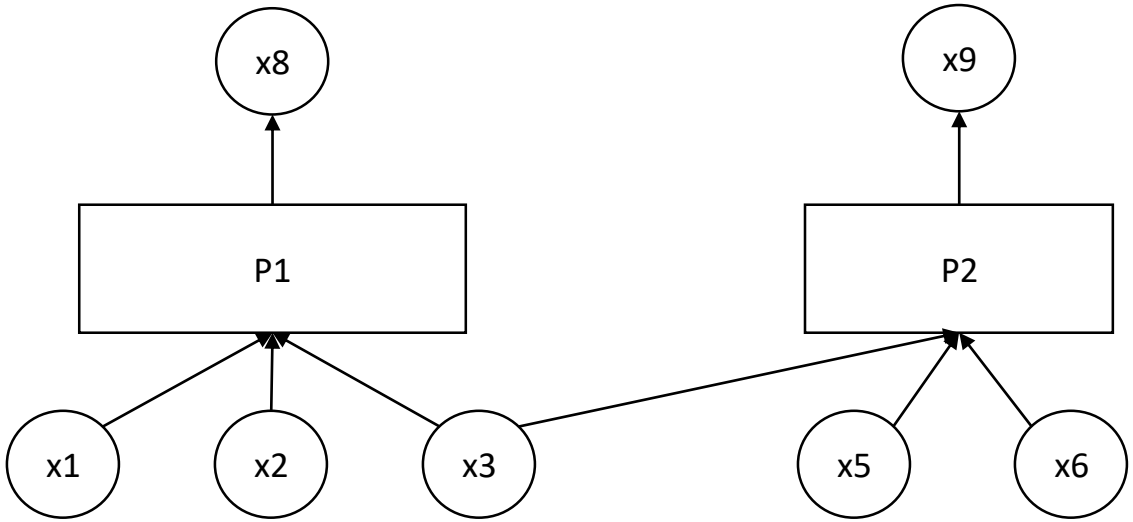


2) Нисходящая часть начинает свою работу в случае возможности получения необходимых результатов.

Все необходимые для вычисления вершины (заданы пользователем) помечаются «нужными», а далее «нужными» помечаются все вершины, без которых невозможно вычислить «нужные».



Вывод плана



std::vector<T> in = {x1,x2,x3,x5,x6}; std::vector<T> req = {x8, x9}

План для x8: x1 x2 x3 P1 x8 x8 = P1(x1,x2,x3)

План для x9: x3 x5 x6 P2 x9 x9 = P2(x3,x5,x6)

Тестирование

```
Name=AreaComputation
Description=Computes area of rectangle from it's width and height.
Type=PythonScript

ComputationalModel=ComputationalModel1

ReceiveMethod=ENV #ENV or Argv

InputParameters=width, height
InputParameterTypes=Int, Int
InputParametersToVariablesMapping=rectangleWidth, rectangleHeight

ResultList=area
ResultTypes=Int
ResultParametersToVariablesMapping=rectangleArea
```

Id:

4 – rectangleArea

1 – AreaComputation

2 – rectangleWidth

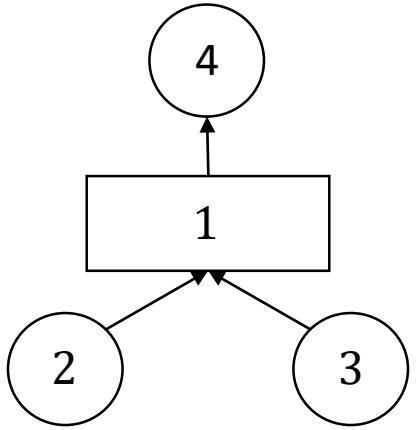
3 – rectangleHeight

Выбрать C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe

```
vertices(g) = 1 2 3 4
edges(g) = (1,4) (2,1) (3,1)

vertex: 1
        out-edges: (1,4)
        in-edges: (2,1) (3,1)
        adjacent vertices: 4
vertex: 2
        out-edges: (2,1)
        in-edges:
        adjacent vertices: 1
vertex: 3
        out-edges: (3,1)
        in-edges:
        adjacent vertices: 1
vertex: 4
        out-edges:
        in-edges: (1,4)
        adjacent vertices:

Plan for computation of variable #4: 2 3 1 4
```



Заключение

Был реализован прототип базы активных знаний, который способен генерировать последовательные планы вычислений.

Данный прототип можно улучшить, добавив возможность параллельного вычисления независимых величин, а также реализовав возможность подстановки раннее вычисленных значений вместо их повторного подсчёта.