Парсер комбинаторы для запросов к графовым базам данных

Курсовая работа

Илья Кириллов,
344 группа

Научный руководитель:
к. ф.-м. н., доц. Григорьев С. В.

Графовые БД

- Вершины -- это записи
- Рёбра -- отношения между записями
- Одна из самых популярных БД -- Neo4j

Парсер комбинаторы

- Способ описать КС-грамматику языка, используя функции высших порядков
- Не требует кодогенерации
- Проверка корректности на этапе компиляции

Библиотека Meerkat

- Библиотека парсер-комбинаторов для языка Scala
- Поддерживает произвольные КС-грамматики

Проблема Context-Free Language Reachability

- КС грамматика
- Граф, помеченный терминалами
- Поиск всех пар вершин таких, что существует путь между ними, описанный данной грамматикой

Постановка задачи

Цель: поддержка графовой БД Neo4j в библиотеке парсер-комбинаторов Meerkat для языка программирования Scala.

Задачи:

- реализовать удобный интерфейс для представления входных данных в виде графа;
- реализовать поддержку работы с вершинами
- реализовать интерфейс для графа Neo4j
- провести экспериментальное исследование

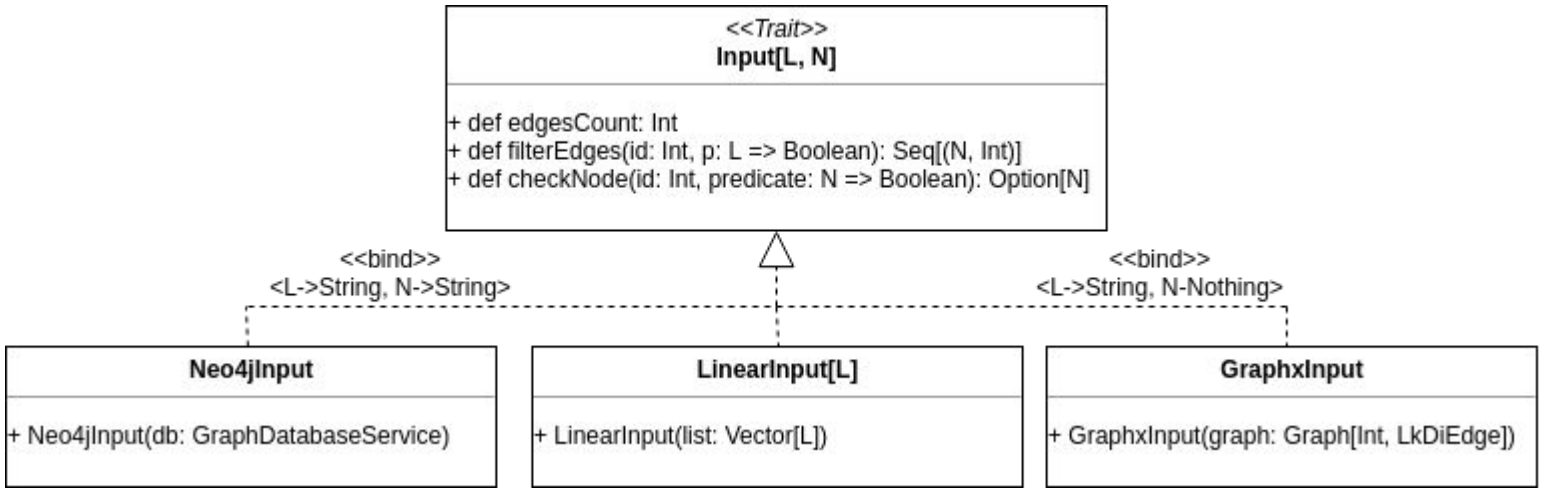
Существующие решения

- Trails -- библиотека комбинаторов
- openCypher -- SQL-подобный язык запросов
- Алгоритм на базе GLL

Предыдущее решение

- слишком большое время работы на графах
- переусложнённый интерфейс входных данных
- невозможность работы с вершинами графа
- невозможность работы с метками, отличными от строк
- невозможность создавать запросы, начинающиеся с произвольной вершины графа.

Интерфейс


<<Trait>>
Input[L, N]
+ def edgesCount: Int
+ def filterEdges(id: Int, p: L => Boolean): Seq[(N, Int)]
+ def checkNode(id: Int, predicate: N => Boolean): Option[N]
<<bind>>
<L->String, N->String>
<<bind>>
<L->String, N-Nothing>
Neo4jInput
+ Neo4jInput(db: GraphDatabaseService)
LinearInput[L]
+ LinearInput(list: Vector[L])
GraphxInput
+ GraphxInput(graph: Graph[Int, LkDiEdge])

Вершины и предикаты

```
val startsWithA = _.name.startsWith("A")
val query =
  V(startsWithA) ~
  E("frend_of")  ~
  V(startsWithA)
```

Экспериментальное исследование

- Проведено тестирование времени работы на бенчмарках для запросов к графам
- Прирост производительности до 2х раз по сравнению с алгоритмом GLL, реализованном в YaccConstructor
- И до 10 раз в случае библиотеки Trails

Экспериментальное исследование

The Same Generation Query

```
val query1 = syn(
  "subclassof-1" ~ query1.? ~ "subclassof" |
  "type-1" ~ query1 .? ~ "type")

val S = syn (
  "subclassof-1" ~ S ~ "subclassof")
val query2 = syn(S ~ "subclassof")
```

Экспериментальное исследование

- Проведено тестирование времени работы на бенчмарках для запросов к графам
- Прирост производительности до 2х раз по сравнению с алгоритмом GLL, реализованном в YaccConstructor
- И до 10 раз в случае библиотеки Trails

Экспериментальное исследование

Онтология	Тройки	Запрос 1					Запрос 2				
		Кол-во результатов	Граф в памяти (мс)	Neo4j (мс)	Trails (мс)	GLL (мс)	Кол-во результатов	Граф в памяти (мс)	Neo4j (мс)	Trails (мс)	GLL (мс)
atom-primitive	425	15454	112	167	2849	232	122	49	52	453	19
biomedical-measure-primitive	459	15156	226	247	3715	482	2871	34	42	60	26
foaf	631	4118	16	25	432	29	10	1	2	1	1
funding	1086	17634	123	152	367	179	1158	18	23	76	13
generations	273	2164	6	21	9	12	0	0	0	0	0
people_pets	640	9472	63	84	75	80	37	2	3	2	1
pizza	1980	56195	544	650	7764	793	1262	44	47	905	50
skos	252	810	4	9	6	6	1	0	1	0	0
travel	277	2499	21	55	34	21	63	2	2	1	2
univ-bench	293	2540	15	43	31	24	81	2	2	2	1
wine	1839	66572	543	727	3156	606	133	5	7	4	5

Для замеров использована библиотека ScalaMeter

Результаты

- реализован интерфейс для представления входных данных в виде графа
- реализована поддержка работы с вершинами и предикатами
- реализована поддержка работы с графовой базой данных Neo4j
- проведено экспериментальное исследование