Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Gráfmintaillesztő rendszerek tesztelése automatikus gráfgenerátorokkal

TDK dolgozat

Készítette:

Bekő Mária

Konzulens:

Semeráth Oszkár

2018

Tartalomjegyzék

Kivonat

Napjainkban az adatokat többféle formátumban is tárolják. Ezek közé tartoznak a gráfadatbázisok, ahol csomópontok reprezentálják az entitásokat és az élek az entitások közötti kapcsolatokat. Az adatstruktúrához illeszkedve többféle gráflekérdező nyelv jött létre, amelyek képesek komplex struktúrák felírására.

A gráfmintaillesztő rendszerek tesztelése azonban komoly kihívást jelent, főképp automatizált megoldásokban nem bővelkedünk. A legnagyobb kihívást ebben az esetben a változatos modellek és lekérdezések automatikus és szisztematikus előállítása jelenti, melyek tesztbemenetként szolgálnak. Továbbá, gráfadatbázisok teljesítménymérését is nagyban segítené az automatikusan előállított modellkészlet.

Dolgozatom célja hogy ezekre a problémákra megoldást találjak. Kutatásom során megmutatom, hogy egy automatikusan előállított diverz modell halmazzal, amelynek modelljei lekérdezésként értelmezhetőek egy gráfmintaillesztő rendszerben (pl.: VIATRA vagy Neo4j), hogyan lehetséges az adott gráfmintaillesztő rendszer tesztelése.

Munkám során fejlett logikai következtetők alkalmazásával állítok elő modelleket, melyek diverzitását szomszédsági formákkal (neighborhood shape-ek) biztosítom. A logikai következtetők eredményeit lekérdezésekként, és adatbázisok tartalmaként egyaránt értelmezhetjük, amelyek eredményei különböző megvalósításokkal összehasonlíthatóvá válnak. A megoldásomat egy esettanulmány keretében prezentálom.

Ezzel a módszerrel lehetővé válik, nagyobb megbízhatóságú gráfmintaillesztő rendszerek fejlesztése olcsóbban. Illetve egy ekkora modell halmaz különböző gráfmintaillesztő rendszerekben lekérdezésekre fordítva és a válaszidőket lemérve teljesítménymérésekre is használható.

Abstract

Nowadays the data is stored in multiple formats. One of this is the graph database, where entities are represented as graph nodes and the relations between entities as edges. Utilizing rich data structure, a variety of graph querying languages have been created in order to query complex structures.

Testing of graph query engines is a challenging task, especially in an automated way. The greatest challenge in this case is the automatic and systematic creation of a diverse set of models and queries, which serve as test inputs. In addition, the development of performance benchmarks for graph databases would be greatly aided.

The purpose of my thesis is to find solutions to these problems. In my thesis, I will show an approach to automatically generate a diverse set of models, that can be interpreted as queries of the graph query engine under test (e.g. VIATRA model query language or Neo4j graph database queries). Additionally, I propose a testing process.

In the course of my work, I produce models with the help of state-of-the-art logic solvers (SAT solvers and Graph Solvers), and use neighborhood shapes to ensure their diversity and create effective equivalence partitioning. The results of the logic solvers are interpreted as queries and as databases content, and the result of query evaluation can be compared to other implementations. I illustrate my solution in a case study.

This method makes it possible to develop more reliable graph query engines at a lower cost. And such a set of models translated to multiple graph querying languages can be used in those graph query engines for performance measurements by measuring response times.

1. fejezet

Bevezetés

Kontextus. Napjainkban ....

Problémafelvetés. Azonban ...

Célkitűzés. Dolgozatom célja ...

Kontribúció. Dolgozatomban bemutatok ...

Hozzáadott érték. Ezáltal ...

Dolgozat felépítése.A második fejezetben bemutatom a dolgozat megértéséhez szükséges háttérismereteket. blabla ...

2. fejezet

Előismeretek

2.1. Esettanulmány

2.2. Gráflekérdező rendszerek

2.2.1. Neo4j

2.3. Modellezés és metamodellezés

2.3.1. Cypher query-k metamodellje

2.3.2. xText

2.3.3. Viatra jólformáltsági kényszerek

2.4. Gráfgenerálás

ábra milyen bemenetek milyen kimenetek

3. fejezet

Áttekintés

3.1. Funkcionális áttekintés

3.2. Blokkdiagram

3.3. A tervezett alkalmazás felépítése

4. fejezet

Gráflekérdezések automatikus generálása

4.1. Lekérdezések generálása

4.1.1. Nyelv minimalizásása

4.1.2. Jólformáltsági kényszerek betartása

Ahhoz, hogy a nyelvtan alapján olyan modelleket tudjunk generálni, amelyeket vissza tudunk majd fordítani szintaktikailag helyes lekérdezésekre, ahhoz kényszereket kell felállítanunk, amelyek betartására kötelezzük a modellgenerátort.

Én a következő kényszereket írtam fel: `pattern x(s:State) State(s);`, ahol a `s` változó jelöli a...

```
pattern goodReferenceToVariable(q : SinglePartQuery, match : Match, vari : VariableDeclaration,  variRef : VariableRe
  SinglePartQuery.readingClauses(q ,match);
  Match.^pattern.patterns.^var(match, vari);
  SinglePartQuery.^return.body.returnItems.items.expression(q , variRef);
  VariableRef.variableRef(variRef, vari);
}

@Constraint(severity = "error", key ={variRef}, message ="error")
pattern notGoodReferenceToVari(variRef : VariableRef){
  neg find goodReferenceToVariable(_, _, vari ,variRef );
  VariableRef.variableRef(variRef,vari);
}
```

A fenti két kényszer azt biztosítja, hogy a lekérdezésnek, amit generálunk mindenképpen meglegyen a két fő része. A match részben határozzuk meg azt a bizonyos mintát, amit keresünk, a return részben pedig visszatérünk az erre a mintára illeszkedő gráfrészletekkel.

Ezen belül ki kell kössük szintén, hogy a két fő rész ne csak csonkként vagy helytelenül generálódjon, hanem az egész elengedhetetlen részfa megszülessen.

Ezért van szükség a return ágon az alábbi kényszerekre:

4.1.3. Diverzitás biztosítása

4.2. Automatizálás

5. fejezet

Értékelés

5.1. Benchmark eredmények

5.2. Feature fedés eredmények

5.3. Diverzitás mérése

6. fejezet

Összefoglalás

6.1. Elméleti eredmények

6.2. Gyakorlati eredmények

6.3. Jövőbeli tervek

Köszönetnyilvánítás

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Irodalomjegyzék