УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Мирко Д. Спасић

МОДЕЛОВАЊЕ УПИТНИХ ЈЕЗИКА СА ПРИМЕНАМА У РЕФАКТОРИСАЊУ И ОПТИМИЗАЦИЈИ КОДА

докторска дисертација

Београд, 2020.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Mirko D. Spasić

MODELING OF QUERY LANGUAGES AND APPLICATIONS IN CODE REFACTORING AND CODE OPTIMIZATION

Doctoral Dissertation

Belgrade, 2020.

Ментор:

др Милена Вујошевић Јаничић, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Ненад Митић, редовни професор
Универзитет у Београду, Математички факултет

др Филип Марић, ванредни професор
Универзитет у Београду, Математички факултет

др Силвиа Гилезан, редовни професор
Универзитет у Новом Саду, Факултет техничких наука

Датум одбране: __.__.2021.

Мајци и оцу

Захвалница

У изради ове докторске дисертације, имао сам безрезервну помоћ блиских особа, својих колега и породице, без којих овај рад не би био потпун. Овом приликом бескрајно им се захваљујем, и наводим оне који су највише утицали да се теза приведе крају и поприми тренутну форму.

Посебну и неизмерну захвалност дугујем својој менторки, проф. др Милени Вујошевић Јаничић, која је својим несебичним залагањем, правовременим коментарима и стручним саветима усмеравала ову дисертацију и активно учествовала у свим фазама њене израде. Желим нагласити и њен утицај на моје раније фазе студирања, почевши од бруцошких дана и вежби из Основа програмирања, где сам на прави начин био упућен у рачунарске науке и када ми је усађена „глад" за овом облашћу која ме и даље држи и временом постаје све јача.

Захваљујем се и члановима комисије, уваженим професорима Ненаду Митићу, Филипу Марићу и Силвии Гилезан, који су детаљним читањем овог рукописа и својим вредним сугестијама значајно допринели коначном обликовању овог рада. Посебно бих истакао позитиван утицај професора Филипа Марића, како на мој научни развој, тако и на наставном плану, где сам имао константу подршку и уживао велико поверење. Веома сам захвалан и професору Предрагу Јаничићу, који је у току израде ове дисертације увек био доступан за консултације и спреман да подели своје знање, штедећи ми драгоцено време. Захваљујем се и својим колегама, члановима Катедре за рачунарство и информатику, који су ми у досадашњој каријери помагали да постанем бољи предавач и истраживач, од којих морам издвојити професора Душка Витаса који ме је на прави начин упутио у наставне и научне воде. Велико хвала упућујем и колегама Орију Ерлингу, Хјуу Вилиамсу и доценту Милошу Јовановићу, који су увек имали разумевања за моје факултетске обавезе и који су ме заинтересовали за област база података, Семантичног веба, повезаних података и упитног језика SPARQL, уједно и истраживачких области ове тезе.

Све ово не би било могуће без огромне подршке моје породице, брата Петра и посебно мојих родитеља, мајке Мирјане и оца Драгослава, којима и посвећујем ову тезу. Коначно, највећу захвалност дугујем супрузи Јели и ћерки Неди, које су ме својом великом љубављу, осмесима и вером у мој успех бодриле свих ових година и умногоме олакшале остварење мојих професионалних циљева. Хвала им на толерисању мојих некада превеликих обавеза, позитивној енергији и безусловном разумевању.

Аутор

Наслов дисертације: Моделовање упитних језика са применама у рефакторисању и оптимизацији кода

Резиме: Проблем садржаности упита један је од фундаменталних проблема у рачунарским наукама, иницијално дефинисан за релационе упите. Са растућом популарношћу SPARQL упитног језика, проблем постаје релевантан и актуелан и у овом новом контексту. У тези је представљен оригинални приступ решавању овог проблема заснован на свођењу на задовољивост у логици првог реда. Подржана је садржаност упита узимајући у обзир RDF схему, а разматра се и релација стапања, као слабија форма садржаности. Доказана је сагласност и потпуност предложеног приступа на широком спектру језичких конструката. Описана је и његова имплементација, у виду решавача SpeCS, чији је кôд јавно доступан. Представљени су резултати детаљне експерименталне евалуације на релевантним скуповима примера за тестирање који показују да је SpeCS ефикасан, и да у поређењу са осталим савременим решавачима истог проблема даје прецизније резултате у краћем времену, уз бољу покривеност језичких конструката. Једна од примена моделовања упитних језика може бити и при рефакторисању апликација које приступају базама података. У таквим ситуацијама, врло су честе измене којима се мењају и упити и кôд на језику у коме се они позивају. Такве промене могу сачувати укупну еквивалентност кода, док на нивоу појединачних делова еквивалентност не мора бити одржана. Коришћење алата за аутоматску верификацију еквивалентности рефакторисаног кода може да дâ гаранцију задржавања понашања програма и од суштинског је значаја за поуздан развој софтвера. Са том мотивацијом, у тези се разматра и моделовање SQL упита у теоријама логике првог реда, којим се омогућава аутоматска провера еквивалентности C/C++ програма са уграђеним SQL-ом, што је и имплементирано у виду јавно доступног алата отвореног кода SQLAV.

Кључне речи: верификација софтвера, SPARQL, садржаност упита, FOL моделовање, SMT решавање, SpeCS решавач, уграђени SQL, рефакторисање кода, SQLAV алат

Научна област: рачунарство

Ужа научна област: спецификација и верификација софтвера

Dissertation title: Modeling of Query Languages and Applications in Code Refactoring and Code Optimization

Abstract: The query containment problem is a very important computer science problem, originally defined for relational queries. With the growing popularity of the SPARQL query language, it became relevant and important in this new context, too. This thesis introduces a new approach for solving this problem, based on a reduction to satisfiability in first order logic. The approach covers containment under RDF SCHEMA entailment regime, and it can deal with the subsumption relation, as a weaker form of containment. The thesis proves soundness and completeness of the approach for a wide range of language constructs. It also describes an implementation of the approach as an open source solver SpeCS. The experimental evaluation on relevant benchmarks shows that SpeCS is efficient and comparing to state-of-the-art solvers, it gives more precise results in a shorter amount of time, while supporting a larger fragment of SPARQL constructs. An application of query language modeling can be useful also along refactoring of database driven applications, where simultaneous changes that include both a query and a host language code are very common. These changes can preserve the overall equivalence, without preserving equivalence of these two parts considered separately. Because of the ability to guarantee the absence of differences in behavior between two versions of the code, tools that automatically verify code equivalence have great benefits for reliable software development. With this motivation, a custom first-order logic modeling of SQL queries is developed and described in the thesis. It enables an automated approach for reasoning about equivalence of C/C++ programs with embedded SQL. The approach is implemented within a publicly available and open source framework SQLAV.

Keywords: software verification, SPARQL, query containment, FOL modeling, SMT solving, SpeCS solver, embedded SQL, code refactoring, SQLAV framework

Research area: computer science

Research sub-area: software specification and verification

Садржај

1 Увод **1**

1.1 Оптимизације на нивоу система за управљање базама података 3

1.2 Оптимизације и рефакторисања на нивоу апликација 4

1.3 Организација тезе и главни доприноси 5

2 Основе, сродни приступи и алати **7**

2.1 Аутоматско доказивање теорема у служби анализе кода 7

2.2 Упитни језици 8

2.2.1 Упитни језик релационих база података — SQL 8

2.2.2 Упитни језик Семантичког веба — SPARQL 9

2.2.3 Семантике упитних језика 10

2.3 Релација садржаности упита 11

2.3.1 Садржаност релационих упита 11

2.3.2 Решавачи проблема садржаности SQL упита 12

2.3.3 Садржаност графовских и SPARQL упита 12

2.3.4 Решавачи проблема садржаности SPARQL упита 13

2.4 Функционална еквивалентност кода 14

2.4.1 Аутоматска анализа императивног кода 14

2.4.2 Аутоматска анализа апликација са приступом базама података . . 15

2.4.3 Еквивалентност програма и регресиона верификација 16

3 Проблем садржаности упита **18**

3.1 Језик SPARQL и проблем садржаности упита 19

3.2 Моделовање конјунктивних упита у језику SPARQL 31

3.2.1 Трансформација упита у формуле 32

3.2.2 Моделовање проблема садржаности упита 38

3.2.3 Сагласност предложеног моделовања 41

3.2.4 Потпуност предложеног моделовања 54

3.3 Подупити као граф обрасци 59

3.4 Моделовање упита са оператором `union` 68

3.5 Моделовање упита са оператором `optional` 71

3.6 Проширења 73

3.6.1 Оператор `diff` . . . 74
3.6.2 Приступ именима графова и рестрикција над њима . . . 77
3.6.3 Проширења израза и услова . . . 83
3.6.4 Аксиоме опште намене и моделовање RDF схеме . . . 87
3.7 Релација стапања . . . 89
3.8 Имплементација и експериментална евалуација . . . 93
3.8.1 Имплементациони детаљи . . . 93
3.8.2 Скупови примера за тестирање коришћени у евалуацији . . . 93
3.8.3 Поређење са релевантним савременим решавачима . . . 98
3.8.4 Експериментално окружење . . . 99
3.8.5 Резултати евалуације на *QC Bench* тест примерима . . . 100
3.8.6 Резултати евалуације на *SQCFramework* тест примерима . . . 102

4 Рефакторисање кода са уграђеним упитима 103
4.1 Предвиђени случај употребе и други сценарији коришћења . . . 104
4.2 Моделовање уграђених SQL упита . . . 107
4.2.1 Подела оригиналног кода у међурепрезентације . . . 108
4.2.2 Опис C/C++ функције: од IRc до FOL формуле . . . 111
4.2.3 Опис SQL упита: од IRsql и схеме базе до FOL формуле . . . 111
4.2.4 Конструкција описа функције . . . 118
4.3 Конструкција услова еквивалентности . . . 119
4.3.1 Провера еквивалентности уграђених *s*-упита . . . 121
4.3.2 Провера еквивалентности уграђених *m*-упита . . . 122
4.3.3 Проширење приступа: `GROUP BY` и агрегатне функције . . . 129
4.3.4 Логички статус моделовања . . . 132
4.4 Имплементација и експериментална евалуација . . . 132
4.4.1 Имплементациони детаљи . . . 132
4.4.2 Скупови примера за тестирање коришћени у евалуацији . . . 134
4.4.3 Експериментално окружење . . . 134
4.4.4 Резултати евалуације . . . 135

5 Закључци и даљи рад 138
5.1 Главни доприноси тезе . . . 138
5.2 Могућа унапређења . . . 140

Литература 142

Биографија 157

Изјаве 158
Изјава о ауторству . . . 158
Изјава о истоветности штампане и електронске верзије докторског рада . . . 159
Изјава о коришћењу . . . 160

Глава 1

Увод

У данашње време окружени смо огромним количинама података из различитих области који стижу до нас све чешће путем апликација. Апликације приступају како традиционалним базама података, тако и другим складиштима. На пример, основни циљ Сематичког веба [104, 106, 115] је да подаци буду доступни на вебу, дефинисани и повезани на начин који осликава њихово значење. На тај начин они су читљиви и рачунарима, али не само за њихово приказивање, већ и за различите облике аутоматизације, интеграције и поновног коришћења у другим апликацијама.

Значај апликација које приступају базама података додатно расте захваљујући све већим областима примене. У данашње време, осим области финансија и пословне сфере, овакве апликације срећу се све чешће и у свакодневном животу. Њихове примене укључују и системе за пренос телевизијских садржаја путем веба (енгл. *online streaming*), играње видео-игара, друштвене мреже, складиштење података у облаку (енгл. *cloud storage*), куповину путем интернета (енгл. *e-commerce*), системе здравствене заштите, прикупљање и анализу података на уређајима са уграђеним рачунаром (енгл. *embedded devices*) у контексту интернета ствари (енгл. *Internet of Things*) [66, 111, 198], и др.

Осим ширења области примене, и број доступних извора података сваким даном све више расте, како у традиционалном релационом моделу, тако и у другим форматима. На пример, у настојању да се оствари целокупна идеја Сематичког веба, број скупова података у RDF формату (енгл. *Resource Description Framework*) [10, 119, 123] убрзано се повећава. Данас, облак повезаних отворених података (енгл. *Linked Open Data Cloud*) региструје више од 1200 RDF извора података [134] из различитих домена, укључујући владин сектор, медије, новинске чланке, географске податке, природне науке, итд. Број извора података био би још већи када би се у разматрање узели и подаци који нису јавни.

Поред константног пораста овог броја, евидентан је и раст величине самих скупова података. Они се у свом животном циклусу обогаћују, унапређују, обједињују са другим скуповима и на тај начин расту [174]. Због свега тога, апликације за приступ подацима и сами системи за управљање базама података суочавају се са великим изазовима у виду сталног пораста захтеване ефикасности, како би обезбедили својим корисницима неопходне податке у жељеном времену.

Због свега тога, програмери свакодневно одржавају већ развијене апликације са циљем веће ефикасности и боље искоришћености ресурса, тј. баве се унапређивањем квалитета кода, трудећи се да очувају његову функционалност. Генерално прихваћена пракса у области рефакторисања и оптимизације подразумева да свака промена буде пропраћена провером да ли је очувано жељено понашање програма. Ова провера најчешће се

врши тестирањем. Међутим, у таквом сценарију, то може трајати дуго и притом, не гарантовати одсуство разлика у понашању између две верзије кода. С друге стране, коришћење алата за аутоматску верификацију еквивалентности кода може да буде мање временски захтевно и може да дâ гаранцију задржавања понашања програма при рефакторисању или оптимизацији. Такође, у сценаријима у којима је тестирање дистрибуирано, или када постоје посебни специјализовани тимови те намене, ови алати могу бити од користи програмерима који ће много брже добити потврду валидности својих промена. Зато, њихово коришћење може бити од суштинског значаја за ефикасан и поуздан развој софтвера.

Проблем одређивања еквиваленције програма је NP-комплетан, и тренутно реална примена алата који га решавају у индустријском окружењу је ограничена. Разлози су ефикасност и једноставност коришћења алата, и пре свега њихова поузданост. Како би њихово коришћење у потпуности заменило тестирање у развојном процесу, алати морају бити сагласни и потпуни, што је врло јак захтев. У супротном, могу се користити као допуна процесу тестирања, пружајући већи степен поузданости у одсуство грешки, а никако као његова замена. Коришћење несагласних и непотпуних алата уместо тестирања може дати лажну сигурност програмерима, и омогућити неким грешкама да не буду уочене, иако би у процесу тестирања оне можда биле детектоване.

У циљу развоја алата за аутоматску верификацију еквивалентности кода у апликацијама које користе упитне језике за приступ базама података, потребно је прецизно моделовати семантику ових језика. Предмет ове докторске дисертације је моделовање основних конструката упитних језика у теоријама логике првог реда, захваљујући којим ће, употребом одговарајућих решавача, бити могуће доношење корисних закључака о упитима. У циљу рефакторисања или оптимизације, у складу са задатом семантиком, подржано је утврђивање еквивалентности и неких других односа између два упита.

Оптимизације се појављују двојако: на нивоу система за управљање базама података, у смислу бржег складиштења, приступа и претраживања, и на нивоу самих апликација које приступају подацима њиховим усавршавањем и рефакторисањем.

1.1 Оптимизације на нивоу система за управљање базама података

Бржи системи за управљање базама података су стари циљ како програмера и администратора, тако и самих корисника ових система, и као такав, у жижи су вишедеценијског интересовања истраживача из разних домена. Поред ефикасних структура података, погодних формата у којима се чувају подаци и брзих алгоритама, један од главних начина за остваривање овог циља лежи у оптимизаторима упита, који трансформишу жељени упит у неки његов еквивалентни облик који се може ефикасније извршити. Овакве трансформације морају очувати еквиваленцију упита, јер некоректна правила оптимизације могу бити неоткривена дуги низ година, и применом могу проузроковати озбиљне проблеме [182, 167]. Еквивалентност се може осигурати њеном верификацијом.

Са аспекта глобалне оптимизације у оквиру статичке анализе упита, најзначајнији проблем је *садржаност упита*, који се разматра у овој дисертацији. То је проблем дефинисан још давне 1977. године, али за упите над релационим базама, као провера да ли сваки резултат једног упита јесте у исто време и резултат другог упита за било коју базу података [38]. Ово је један од фундаменталних проблема у области база података и у општем случају је неодлучив. За неке класе упита овај проблем је NP-комплетан и представља велики изазов за истраживаче.

Опис проблема природно се преноси на језик SPARQL (енгл. *Simple Protocol and RDF Query Language*), упитни језик и протокол за приступ подацима [155, 156]. Иако је језик релативно нов, већ има истраживача који се баве овим проблемом баш у овом језику [44, 151]. Управо је то и тема једног дела ове докторске дисертације. Исправна оптимизација упита у овом језику јесте фундаментално значајна за постизање практичне корисности великих количина података доступних у RDF формату. Овоме сведочи и велики број дизајнираних скупова примера за тестирање (енгл. *benchmarks*) како би се поспешио даљи развој триплет-складишта (енгл. *triplestores*) и њихових оптимизатора [4, 93], као и генератора синтетичких скупова података погодних за евалуацију [176, 184].

Други значајни проблеми, као што су *еквивалентност упита* и *задовољивост упита*, могу се свести на проблем садржаности упита. Два упита су *еквивалентна* ако имају једнаке скупове одговора над било којом базом, што се своди на проверу задовољења релације садржаности упита у оба смера, тј. ако је први упит садржан у другом, и обратно. Ако су два упита еквивалентна, оптимизатор може извршити један упит уместо другог, у случајевима када такав избор води ефикаснијем извршавању.

Решавачи проблема садржаности упита могу се користити и за проверу задовољивости упита (упит је *задовољив* ако постоји бар једно његово решење), свођењем проблема на утврђивање еквиваленције (или садржаности) упита са неким упитом за који је познато да нема решења. Ако упит није задовољив, и ако се незадовољивност може закључити статички, без његове евалуације (извршавања), то може уштедети значајно време, посебно у дистрибуираним окружењима (што је најраспрострањенији сценарио данас), када се уједињени упити (енгл. *federated queries*) извршавају над више различитих извора података преко потенцијално споре мреже [203].

У релационом контексту постоје додатне примене решавача проблема садржаности, као што су у техникама интеграције [33, 188] и прикупљања информација [85, 127], провере интегритета ограничења [78, 101, 100], и репрезентације доменског знања заснованог на дескриптивној логици или графовској репрезентацији [34, 71, 72, 128]. Све ове примене могу се природно пренети у контекст Семантичког веба и језика SPARQL.

1.2 Оптимизације и рефакторисања на нивоу апликација

Упитни језици користе се најчешће у комбинацији са неким програмским језиком опште намене. Резоновање о таквим програмима постаје далеко компликованије, јер укључује семантике и императивних и декларативних језика. Примена моделовања упитних језика могућа је и у овим сценаријима, што се у дисертацији и разматра на примеру SQL језика, као најраспрострањенијег упитног језика за приступ системима за управљање релационим базама података [88]. Овај језик постао је стандард у овој области, и као такав неизоставни је део многих апликација и велики број програмера га свакодневно користи. Из тог разлога, сваки помак у смислу унапређивања процеса развоја овог типа апликација је врло значајан, нарочито ако га убрзавају, олакшавају и чине поузданијим.

Додатни фокус ове тезе је један такав корак, на примеру SQL језика уграђеног у језик C/C++ (енгл. *embedded* SQL). На тај начин, омогућено је аутоматско резоновање о функционалној еквивалентности C/C++ програма који садрже уграђене SQL упите за приступ подацима, у циљу једноставнијег рефакторисања и оптимизације такве врсте кода. Приступ је заснован на иновативном моделовању понашања програма и модерним алатима за формално резоновање [17, 15]. Значајност овог приступа образлежена је у наставку.

Спроведена анкета [116] која је укључивала 328 *Microsoft* инжењера (од којих су већина били програмери, а остали су припадали тестерима, архитектама и програм менаџерима) показала је да рефакторисање подразумева знатне трошкове и ризике, као и да су програмерима потребни различити алати као подршка овим активностима. Скоро трећина обухваћених у анкети жалила се на недостатак таквих алата. Пуно истраживачког и програмерског труда уложено је у развој алата за рефакторисање [143], како би се ове активности аутоматизовале колико је могуће и повећале продуктивност програмера [135, 84]. Такође, већина испитаника нагласила су да је један од најбитнијих изазова у вези са рефакторисањем неизбежне потешкоће у осигуравању исправности програма након завршеног процеса.

У пракси се потврдило да је велики број рефакторисања обавезно праћен већом учестаношћу пријаве грешака у програмима [200, 201]: некомплетна или нетачна рефакторисања проузрокују грешке [96]. Зато, такве активности треба да буду подржане регресионом анализом (регресионим тестовима и регресионом верификацијом), јер свака промена у коду може ненамерно проузроковати нове грешке. Коришћење аутоматске провере еквивалентности може скратити потребно време јер се у идеалном случају може користити уместо тестирања, или као његова допуна отпорна на грешке проузроковане људским фактором (као што је недостатак случаја за тестирање), па се стога сматра много поузданијом методом од тестирања [157]. У остатку текста, израз рефакторисање биће коришћен у ширем смислу, тј. под рефакторисањем подразумеваће се све промене кода које чувају функционалну еквивалентност, укључујући нпр. промене које су уведене из оптимизационих разлога.

1.3 Организација тезе и главни доприноси

Остатак тезе организован је по следећим главама:

Глава 2 садржи кратак преглед релевантних информација и општих појмова који се користе у наставку тезе. Дата је функција аутоматског доказивања теорема у служби анализе кода, представљени су упитни језици који се користе у тези, SQL и SPARQL, и њихове семантике. Дефинисан је проблем садржаности упита у ова два језика, и дат је преглед алата који их решавају. Такође, глава даје преглед аутоматске анализе кода, императивног, упитног и њихове комбинације, као и еквивалентности програма, тј. регресионе верификације.

Глава 3 формално представља синтаксу и семантику језика SPARQL и дефинише проблем садржаности упита у овом језику. Представљено је моделовање великог броја језичких конструката и моделовање проблема садржаности упита у теоријама логике првог реда. Овим је омогућено ефикасно аутоматско резоновање о овом проблему коришћењем одговарајућих решавача. Доказана је сагласност и потпуност предложеног моделовања и за релацију садржаности упита у својој стандардној форми и за релацију стапања, која је често коришћена форма релације садржаности. На крају главе, приказани су и резултати евалуације имплементираног алата у поређењу са савременим решавачима на већ постојећим скуповима примера за тестирање.

Глава 4 представља још једну примену моделовања упитних језика у теоријама логике првог реда. Повезују се семантике императивних и упитних језика, и на тај начин омогућава аутоматско резоновање о функционалној еквивалентности императивних програма који садрже приступ релационим базама података кроз уграђене SQL упите. На тај начин, унапређује се процес рефакторисања оваквог типа кода, пружајући додатну сигурност програмерима при овој свакодневној активности.

Глава 5 излаже главне теоријске и практичне закључке тезе, и представља могуће правце даљих истраживања који би се ослањали на резултате приказане у овој дисертацији.

Главни доприноси тезе су:

- Оригинално моделовање основних конструката упитних језика SPARQL (поглавља 3.2, 3.3, 3.4, 3.5 и 3.6) и SQL (поглавље 4.2) у теоријама логике првог реда, које се користи у различитим сценаријима, и то при:
  - моделовању проблема садржаности, еквиваленције и задовољивости упита у језику SPARQL (поглавље 3.2.2), за које је доказана сагласност (поглавље 3.2.3) и потпуност (поглавље 3.2.4), и које се може користити за проверу еквивалентности упита насталих презаписивањем у оквиру оптимизатора упита,
  - моделовању проблема функционалне еквивалентности сегмената императивног кода који садржи приступ релационим базама података користећи уграђене SQL упите, које се може користити за утврђивање еквивалентности оригиналног и рефакторисаног кода (поглавље 4.3);

- Имплементација јавно доступних алата отвореног кода за резоновање о овако моделованим проблемима:
  - алат SpeCS, сагласан и потпун решавач проблема садржаности упита у језику SPARQL, надмашује остале савремене приступе у решавању овог проблема, како у погледу ефикасности, тако и са аспекта подржаности различитих језичких конструката (поглавље 3.8),
  - алат SQLAV, користан у процесу рефакторисања императивног кода са уграђеним SQL упитама, доказује еквиваленцију две верзије кода или указује на могуће пропусте и грешке (поглавље 4.4);
- Унапређивање постојећих корпуса за проверу коректности и развијање нових, који су расположиви и корисни свим истраживачима који се баве овом проблематиком:
  - откривање и отклањање проблема у постојећим скуповима тестова садржаности SPARQL упита (поглавље 3.8.2),
  - корпус са више стотина примера рефакторисања C/C++ функција са уграђеним SQL упитима, изграђен за потребе евалуације развијеног алата SQLAV, и у будућности сличних алата (поглавље 4.4.2).

Глава 2

Основе, сродни приступи и алати

У овом поглављу дат је преглед свих неопходних општих информација о језицима који се користе у тези, њиховим семантикама, сродним методама, техникама, приступима и алатима који решавају сличне проблеме.

Преглед главе: У поглављу 2.1 дата је веза између аутоматског доказивања теорема и анализе кода. Научна истраживања сродна приступу који је приказан у овој тези могу се поделити у две велике групе: она која се баве искључиво упитним језицима (кратак преглед релевантних упитних језика дат је у поглављу 2.2), тј. разним односима између два упита, на пример садржаношћу, еквиваленцијом и задовољивошћу упита (поглавље 2.3), и она чији је предмет изучавања функционална еквивалентност кода (поглавље 2.4), и то пре свега императивног, као најстарије и најистраживаније парадигме.

2.1 Аутоматско доказивање теорема у служби анализе кода

Врло често, у циљу разматрања било каквих својстава императивних и декларативних програма, она се моделују прецизним математичким језиком, тј. формулама у одговарајућим математичким теоријама. Уместо ручног доказивања важења моделованих својстава од стране експерата, користе се већ развијени ефикасни доказивачи и решавачи, који овај посао аутоматизују, скраћују његово потребно време и гарантују већу поузданост у спроведене доказе. Управо је верификација софтвера (и хардвера) главна област примене ових алата.

FOL доказивачи опште намене заснивају се уобичајено на униформним процедурама доказивања као што је метод резолуције, и типични представници ове класе су *Vampire* [158] и *Spass* [199], од којих ће се првонаведени и користити у евалуацији приступа и алата развијених у оквиру ове тезе.

Решавачи задовољивости у односу на неку теорију (енгл. *Satisfiability Modulo Theories solvers*), или скраћено SMT решавачи, проверавају задовољивост буловске комбинације услова формулисане у некој одређеној теорији логике првог реда, или комбинацији таквих теорија. Често коришћена комбинација је линеарна аритметика (енгл. *linear arithmetic*) са једнакостима, неинтерпретираним функцијама и низовима, која се у даљем тексту означава са **L**. Уместо линеарне аритметике, може се користити бит-вектор аритметика (енгл. *bit-vector arithmetic*), и тада се одговарајућа комбинација означава са

B. Развој SMT решавача вођен је њиховом применом у верификацији софтвера и хардвера. Неки решавачи, укључујући и *Z3* [67], могу резоновати и са специфично додатим аксиомама које проширују стандардне одлучиве теорије над којим се резонује. Комуникација са решавачем може се реализовати кроз његов API или користећи екстерне датотеке у SMT-lib формату [16].

2.2 Упитни језици

За прављење упита над над базама података и информационим системима користе се упитни језици (енгл. *query languages*). Њихов циљ може бити проналажење конкретних одговора на постављена питања (упите) приступањем бази података, или проналажење докумената који садрже релевантне одговоре, као што је то случај са упитним језицима претраживача[1] које користе Гугл (енгл. *Google*) или Бинг (енгл. *Bing*). У овој тези акценат је на упитним језицима за приступ базама података, и то језицима SQL и SPARQL.

2.2.1 Упитни језик релационих база података — SQL

Упитни језик SQL (енгл. *Structured Query Language*) је широко распрострањен и усвојен као језик за чување, обраду и приступ подацима у релационим базама података [88]. Настао је седамдесетих година у лабораторији компаније IBM из већ постојећих релационих језика *SQUARE (Specifying QUeriesAs Relational Expressions)* [29] и *SEQUEL (Structured English QUEry Language)* [36], који припадају декларативној парадигми, тј. њима се описује шта се жели добити, без прецизирања начина на који се то може реализовати.

Језик је први пут стандардизован још 1986. године, а актуелна верзија стандарда која прецизира његову синтаксу и семантику може се наћи на вебу [110]. Уз претпоставку да су познати основни конструкти језика и основе релационог модела података, може се написати упит из примера који следи.

Пример 2.1 Доступна база садржи податке о студентима (табела `students`, са атрибутима `id` који представља њихов јединствени идентификатор, `name` који садржи име студента, `type` који прецизира да ли се ради о студенту основних, мастер или докторских студија, итд.), испитима које они слушају (табела `courses` са атрибутом `name` који садржи име испита, итд.), и везама између ове две табеле (`student_course`) које чувају податке о изабраним испитима за сваког студента. Упит $\texttt{Q}_1$ приказан на слици 2.1 (лево) враћа идентификаторе и имена студената који слушају курс чије је име `Programming1`.

Језик SQL је и даље јако актуелан. Према *StackOverflow* анкетама[2], 54.4% програмера користи овај језик, који се и даље сматра доминантним упркос новим трендовима база података [163]. На пример, узимајући у обзир само пројекте присутне у *GitHub* репозиторијумима, више од 473 хиљаде јавних репозиторијума[3], са више од 144 милиона линија кода, користе SQL.

[1] Синтакса оваквих упитних језика је проста, а семантика сиромашна, па их многи и не сматрају упитним језицима.

[2] Резултати анкете програмера за 2019 годину, доступни на адреси `https://insights.stackoverflow.com/survey/2019/`, садрже одговоре 87.354 програмера.

[3] Укључујући основне и одвојене (енгл. *forked*) репозиторијуме.

Овај језик може се користити интерактивно, у склопу наменског алата, покретањем *ad-hoc* упита или административних задатака над базом. Међутим, његово уобичајено коришћење је кроз апликације, где се операције над базом података позивају кроз програме написане на програмском језику опште намене, тј. матичном језику (енгл. *host language*), као што су C/C++, *Java*, *JavaScript*, *PHP*, и др. Више од 65% свих *GitHub* репозиторијума који користе SQL су такође означени са једним додатним језиком опште намене. Узимајући у обзир само 100 најважнијих *GitHub* репозиторијума, 76% њих користи SQL (у комбинацији са другим језицима опште намене) [89].

За позивање SQL наредби из апликација, користи се API (енгл. *application programming interfaces*) или уграђени SQL. У првом приступу, систем за управљање базом података обезбеђује своју библиотеку (енгл. *native* API) или се може користити независна библиотека, као што је ODBC [28] или JDBC [79]. Код другог приступа, SQL наредбе могу се мешати са наредбама матичног језика. Сви системи за управљање релационим базама морају подржавати уграђивање SQL упита у језик опште намене. На пример, *Altibase* [2], *Microsoft SQL Server* [139, 20], *Oracle Database* [146, 30], *PostgreSQL* [154, 145], *SAP Sybase* [183, 87] и многи други подржавају уграђивање SQL упита у C/C++ програме.

SQL упит Q_1

```sql
select t1.id, t1.name
from students as t1,
    student_course as t2,
    course as t3
where
    t1.type = "UndergradStudent"
    t1.id = t2.s_id and
    t2.c_id = t3.id and
    t3.name = "Programming1"
```

SPARQL упит Q_2

```
select ?x ?n
where {
    ?x a :Student .
    ?x a :UndergradStudent .
    ?x :name ?n .
    ?x :takesCourse ?y .
    ?y a :Course .
    ?y :name "Programming1"
}
```

Слика 2.1: Пример SQL упита Q_1 (лево) и њему одговарајући пример SPARQL упита Q_2 (десно)

2.2.2 Упитни језик Семантичког веба — SPARQL

Упитни језик SPARQL је новијег датума. Његов први стандард је објављен 2008. године [156], и од тада је препознат као једна од кључних технологија Семантичког веба. Он представља декларативни семантички упитни језик за базе података чији су подаци похрањени у RDF формату, тј. као субјекат-предикат-објекат триплети. Детаљније информације и више примера о моделу података и о самом језику, његовој синтакси и семантици, дате су у поглављу 3.1.

За разлику од језика SQL којим се приступа подацима у релационом моделу, тј. где упити зависе од схеме базе података, језик SPARQL приступа моделу података без схеме (енгл. *schemma-less*). То га чини идеалним у контексту Сематичног веба, где се подаци најчешће налазе у различитим изворима које треба објединити, што је много изводљивије ако структура тих података није ограничена схемом, као у релационом моделу.

Пример 2.2 На слици 2.1 (десно) наводи се пример SPARQL упита који одговара SQL упиту са леве стране исте слике. Како је модел података без схеме, нема `from` клаузуле у SPARQL упиту којом се прецизирају табеле у схеми података код SQL упита, већ се може претпоставити да се сви подаци о студентима, испитима и о њиховим везама налазе на једном месту. Слично као у случају SQL упита, клаузула `select` садржи пројекције, а клаузула `where` садржи услове које подаци треба да задовоље.

У терминима релационог модела података, RDF формат може се представити једном табелом која има три колоне, по једну за субјекат, предикат и објекат. На тај начин, систем за управљање релационим базама података може се користити и у овом новом контексту, али је потребно превести SPARQL упите на одговарајуће у SQL језику [175, 114]. Обратно, ако би RDF формат података требало представити релационим моделом, субјекти би одговарали самим редовима табела, тј. атрибутима примарних кључева, предикати атрибутима табела, док би објектима одговарале вредности тих атрибута, тј. одговарајуће вредности атрибута редова који одговарају субјекту тог триплета.

Језик је базиран на преклапању графовских образаца (из клаузуле `where`) са RDF графовима (подацима у бази). Осим што представља упитни језик, SPARQL дефинише и протокол употребе упитног језика преко HTTP протокола и формат његових резултата. Растом популарности Семантичког веба, и све већим бројем доступних скупова података у RDF формату, расте и заступљеност овог језика у пракси, али и заинтересованост истраживача са теоријског аспекта.

2.2.3 Семантике упитних језика

Семантике програмских језика баве се проучавањем значења програмских језика, тј. синтаксно исправним конструкцијама језика придружују прецизно значење. Семантика се најчешће дефинише описом процеса извршавања програма и везама између улаза и излаза програма, тј. креирањем модела извршавања (енгл. *model of computation*). Иако то може бити неформално и на природном језику, у семантикама се могу користити и строга математичка правила у прецизно дефинисаним теоријама, како значење језичких конструката не би било вишесмислено и подложно различитим тумачењима.

У контексту упитних језика, семантике прецизно дефинишу резултате евалуације, тј. извршавања, упита над задатом базом података. Осим разумевања карактеристика самог упитног језика и смерница при конструкцији система за управљање базама података који подржава тај језик, семантика је корисна и у доказивањима неких својстава језика. На пример, формалне семантике налазе једну од својих примена у верификацији оптимизатора упита [125, 166]. Врло је важно резоновати о еквивалентности упита формално, јер нетачна правила оптимизације могу у току дугог низа година проузроковати велике штете у различитим доменима [167, 182]. Резоновање о било каквим својствима упитних језика јако зависи од коришћене семантике.

Постоје различите формализације SQL семантике: скуповна семантика (енгл. *set semantics*) [38, 144, 160], мултискуповна семантика (енгл. *bag semantics*) [40, 58, 109] и хибридни приступи (енгл. *bag-set semantics*) [46, 56, 57, 112]. Скуповна семантика дефинише резултат упита као скуп енторки, док мултискуповна дозвољава дупликате, тј. резултат упита у овом случају је мултискуп енторки. У фокусу ових радова посвећених семантикама су теоријски аспекти, док је имплементација сведена на примену процедуре јурења (енгл. *chase procedure*) над конјунктивним упитима [21, 69]. SQL алгебра [22, 23] је развијена са циљем формализације SQL семантике у оквиру алата *Coq*

[24], док се семантике назване HoTTSQL [47] и семантике U-полупрстенова [51] користе за аутоматизацију доказивања еквиваленције у истом алату. Семантике засноване на релационој алгебри [11, 35, 99, 132] су скуповне семантике и нису много повезане са реалним SQL-ом.

Постоје две класе семантика језика SPARQL [155, 149], за које је доказано да су еквивалентне (на конструктима које подржавају) [5], а које се могу разматрати при доказивању различитих својстава језика. Прва, првенствено намењена програмерима, развијена је у оперативном стилу, и први пут стандардизована 2008. године [156]. Скуповне [149] и мултискуповне [6] семантике припадају другој класи семантика, дефинисане су прецизније користећи математичке теорије скупова и мултискупова, али укључују само подскуп језика SPARQL, тј. ове семантике обухватају само најбитније синтаксне конструкције. Касније, проширене су неким комплекснијим језичким конструктима, нпр. подупитима [8, 152], операторима разлике, пројекцијама [7, 6] и негацијама [65].

Једна од примена семантика упитних језика у овој тези јесте управо при разматрању релације садржаности упита, што је и садржај следећег поглавља.

2.3 Релација садржаности упита

Проблем садржаности упита дефинисан је иницијално за релационе базе података и SQL језик још давне 1977. године [38]. Од тада, многи важни теоријски резултати и правци истраживања потекли су од садржаности релационих упита. Са растућом популарности Семантичког веба, разматрање садржаности упита у језику SPARQL такође је постала важна и атрактивна област која привлачи истраживаче широм света.

2.3.1 Садржаност релационих упита

Проблем садржаности упита припада класи неодлучивих проблема [187]. Током неколико последњих декада, истраживачи су активно радили на идентификовању његових одлучивих фрагмената [38, 118, 138, 160].

Проблем је разматран у различитим контекстима, укључујући подршку за различите језичке конструкте, као што су пројекције, спајања, позитивне селекције и уније [1], као и изразе који садрже оператор разлике [160]. Битни контексти такође укључују присуство услова инклузије зависности [113] и специјалне класе упита, као што су позитивни конјунктивни упити [37], као и упити који садрже погледе [70, 45], груписања и функције агрегације [129].

Скорија истраживања изучавају сличне проблеме коришћењем техника машинског учења [103], где се разматра стопа садржаности (енгл. *containment rate*) упита у неком другом за конкретан садржај базе података.

2.3.2 Решавачи проблема садржаности SQL упита

Имплементирање алата за проверу садржаности упита који се може ефикасно користити у пракси је врло захтевно [159]. Како се проблем садржаности најчешће користи за проверу важења релације еквивалентности два упита, то се модерни алати и конструишу управо са тим циљем. У последњих пар година, развијено је неколико таквих алата.

COSETTE [52, 48] је аутоматски доказивач који може закључити еквивалентност два SQL упита. У случају потврдног одговора, алат враћа доказ еквивалентности проверен у алату *Coq* [24], док у супротном, генерише се контра-пример у терминима улазних табела које фигуришу у упитима. Подржава конјуктивне и корелисане упите, спољашња спајања и агрегатне функције. Алат је отвореног кода, доступан у оквиру *GitHub* репозиторијума [50], а могуће му је приступити и преко јавно доступне веб апликације [49].

UDP [51] је доказивач еквивалентности SQL упита заснован на алгебарском приступу. Еквивалентност SQL упита се открива њиховим превођењем у U-изразе (енгл. *U-expressions*), применом правила презаписивања истих и тражењем изоморфизама над њима. У алату се користи доказивач *Lean* [142]. Разматра се присуство индекса, погледа и других услова зависности. Доказана је сагласност приступа имплементираног у оквиру овог алата, као и његова потпуност при разматрању уније конјуктивних упита.

EQUITAS [206] је алат који, за разлику од претходна два, не користи алгебарски приступ, већ имплементира симболичку репрезентацију упита, коју су аутори развили управо за ову сврху. Проблем еквивалентности упита своди се на проблем задовољивости формуле првог реда у односу на једну или више теорија, за чије решавање се користи SMT решавач *Z3*. Подржани су и сложенији упити који садрже агрегатне функције и различите типове спољашњег спајања.

SPES [205] имплементира хибридни приступ који комбинује алгебарски приступ и симболичку репрезентацију упита, и на тај начин решава проблеме појединачних приступа. У првој фази упити се преводе у новоразвијену алгебарску репрезентацију, која се затим нормализује у циљу смањења структурних разлика. У другој фази примењује се верификациони алгоритам који преводи нормализоване репрезентације у симболичке, и доказује њихову еквиваленцију коришћењем SMT решавача.

2.3.3 Садржаност графовских и SPARQL упита

Последњих година, графовске базе података стичу све већу популарност и њихови упитни језици постају активна тема истраживања, а посебно проблеми повезани са садржаношћу упита у овим језицима. На пример, проблеми садржаности, еквивалентности и задовољивости разматрају се за XPath упите који се користе за приступ графовским базама података [81, 97, 102, 121, 140, 141, 168]. Постоје истраживања и у оквиру других, мање популарних, упитних језика [34, 82, 120]. Ипак, на пољу садржаности графовских упита најзаступљенија су истраживања у оквиру језика SPARQL.

Језичка изражајност (енгл. *expressiveness*) језика SPARQL иста је као изражајност релационе алгебре [5]. Према томе, проблем садржаности у језику SPARQL такође није одлучив, али, као у релационом случају, за неке подскупове упита, ипак јесте [169]. На

пример, општа анализа комплексности проблема садржаности и еквиваленције за различите фрагменте језика SPARQL (који покривају широк спектар језичких конструкта, као што су `.`, `union`, `optional` и пројекције) показала је њихову одлучивост и NP-комплетност, али и неодлучивост за неке друге подскупове језика [151]. Проблем садржаности упита разматра се и у својој слабијој форми, тј. у релацији стапања (енгл. *subsumption*), која представља уопштење релације садржаности [9, 125, 149, 151].

Слични проблеми, укључујући еквивалентност граф образаца, који представљају основни конструкт језика SPARQL, такође се разматрају [148, 149, 166]. Њима се приступа идентификујући нормалне форме образаца и испитујући правила трансформација између њих која чувају еквиваленцију. Главну корист оваквих истраживања имају оптимизатори упита, који примењују правила презаписивања (енгл. *rewriting rules*), мењајући оригинални граф образац еквивалентним, чије је извршавање над конкретном базом, тј. евалуација ефикаснија. Таква правила могу се заснивати на одређеним својствима SPARQL оператора [149, 166], или се могу посматрати као правила трансформисања над дрволиким структурама образаца, која се тумаче као планови извршавања упита, сводећи упит у простије форме [125], чије је извршавање PSPACE комплетно [149].

Процедуре за одређивање садржаности SPARQL упита примењују се такође у оптимизацији семантичких упита у превођењу са језика SPARQL на SQL, имплементираном у *Ontop* алату [32], као и у оквиру алата *OnGIS*, који има улогу посредника у евалуацији геопросторних упита [170].

2.3.4 Решавачи проблема садржаности SPARQL упита

Постоји неколико савремених решавача проблема садржаности SPARQL упита:

SPARQL-Algebra (SA) [125] је решавач за проблем провере релације стапања и еквивалентности. Заснива се на дрволикој структури SPARQL упита, названој дрво обрасца, која се може сматрати планом извршавања упита [126]. Имплементиран је над неоптимизованом верзијом SPARQL процесора ARQ [83], додајући правила за трансформације граф образаца, подршку за нормалне форме образаца и могућност тестирања релација између два упита.

Alternation Free two-way μ-calculus (AFMU) [42] решава проблем садржаности упита сводећи га на проблем задовољивости формуле у фрагменту μ-рачуна [122]. Представљени су различити приступи трансформисању упита у формуле овог типа [43, 44], и за сваки од њих користи се исти алат за испитивање задовољивости [185].

TreeSolver (TS) [92] имплементира процедуру за утврђивање садржаности, еквивалентности, преклапања и других релација над NoSQL упитима, као што су XPath [59] или Jaql [26], такође свођењем на проблем задовољивости формуле у μ-рачуну [90, 91]. Овај решавач може се такође користи при провери различитих својстава било које дрволике структуре, укључујући SPARQL упите.

Jena-Sparql-Api Graph-Isomorphism based query containment solver (JSAG) [180] је решавач чији је приступ провера одоздо нагоре поклапања дрволиких структура алгебре упита. Он проверава да ли постоји изоморфизам између нормализованог стабла алгебре израза подупита и подстабла надупита.

Евалуација ових решавача, њихове детаљне карактеристике и поређење са приступом који је развијен у овој тези приказане су у поглављу 3.8.

2.4 Функционална еквивалентност кода

Акценат ове тезе биће на функционалној еквивалентности кода упитних језика, како самосталног, тако и у контексту упита у оквиру императивних програма. У овом делу дат је кратак преглед појмова, релевантних техника и алата који се баве баш тим или неким повезаним проблемима у некој од ове две поменуте парадигме, тј. функционалном еквивалентношћу императивног кода и/или кода упитних језика.

2.4.1 Аутоматска анализа императивног кода

Алати за аутоматску верификацију софтвера настоје да аутоматски провере одређена својства програма. Имајући у виду да је императивна парадигма најстарија и најзаступљенија, највише пажње истраживача привлачила је аутоматска верификација баш императивних програма.

Постоје различити приступи за аутоматску статичку анализу (енгл. *static analysis*), као што су симболичко извршавање (енгл. *symbolic execution*) [117], провера модела (енгл. *model checking*) [55] и апстрактна интерпретација (енгл. *abstract interpretation*) [64]. На овим приступима заснован је и велики број развијених алата.

На пољу симболичног извршавања, истакнутији су алати KLEE [31] и PEX [186], који служе за аутоматско проналажање грешака у коду и генерисање тест примера. Први анализира LLVM кôд уз помоћ SMT решавача *STP* [86], а други се може користити за све језике .NET платформе, и за проверу услова исправности користи алат *Z3* [67]. Проверавањем модела баве се рецимо алати CBMC [54], ESBMC [60], LLBMC [137]. Они анализирају C/C++ програме, и у њима проверавају прекорачења бафера, безбедност у раду са показивачима, поткорачење и прекорачење у аритметичним изразима, кориснички задате услове, и др. У алате чији је приступ апстрактна интерпретација спадају Astrée [27], Polyspace [68] и Coverity SAVE [25]. Они представљају комерацијалне алате и користе се за C-олике програмске језике.

Алат LAV

Како се у имплементацији идеја ове тезе користи алат LAV [193, 195, 191], о њему је дато више речи у наставку. Ради се о алату опште намене заснован на LLVM оквиру [124], који комбинује симболичко извршавање, проверу модела, и SAT кодирање контроле тока програма за конструисање услова коректности, а потом користи SMT решаваче за проверу ових услова. LAV симболички извршава блокове кода и генерише изразе над променљивама датим у виду њихових вредности на почетку блока. Блокови су повезани исказним променљивама које одговарају прелазима између блокова и њиховом извршавању. Могуће везе између различитих блокова дате су одговарајућим условима заснованим на инструкцијама гранања.

LAV може моделовати петље на два начина:

- уопштењем полазног кога (енгл. *over-approximation*) и
- сужавањем опсега полазног кода (енгл. *under-approximation*).

При уопштењу полазног кода, симулира се првих n и последњих m итерација петље. У том случају, после првих n размотавања петље, LAV поништава вредност сваке променљиве које се мења у некој наредби петље (слично као што се ради, нпр. у [14]).

Ресетовањем тих вредности, сва могућа извршавања петље се симулирају, али овај приступ може проузроковати губитак прецизности, у смислу резултујућих лажних упозорења, који се може третирати додатним техникама [193]. Извршавање се даље наставља помоћу m додатних размотавања петље. Ако је уопштење полазног кода без грешака (енгл. *bug-free*), онда то важи и за оригинални кôд.

При сужавању опсега полазног кода, симулира се размотавање петље фиксирани број пута, као што је случај код провере ограничених модела (енгл. *bounded model checking*). Овај приступ нема лажних упозорења, али може пропустити неке проблеме у коду.

LAV може моделовати рачунање у програмском језику помоћу теорије линеарне или бит-вектор аритметике. Теорија неинтерпретираних функција се користи за моделовање неких својстава показивача и теорија низова за моделовање меморије. Алат је отвореног кода (енгл. *open source*) и јавно је доступан [196].

2.4.2 Аутоматска анализа апликација са приступом базама података

Симболичко извршавање над SQL кодом може се користити и у контексту аутоматског генерисања тест случајева (енгл. *automated test case generation*) [76, 189, 147, 130, 131, 94].

Алат описан у [76] реализује конколично (енгл. *concolic*) извршавање да би генерисао тест случајеве, тј. комбинује конкретно и симболичко извршавање кода, па зато подржава анализу динамички генерисаних упита. Qex [189] имплементира симболичко извршавање параметризованих SQL упита, али не разматра рачунања над улазним и излазним параметрима која се могу обављати у матичном језику. SynDB [147] је оквир заснован на трансформисању ограничења наметнутих у `WHERE` клаузули у императивни програмски кôд. Алат описан у [130] имплементира симболичко извршавање над ограниченим подскупом програмског језика *Java* и подржава SQL наредбе и трансакције. Већина алата ради са *Java* кодом као матичним језиком за SQL и сви осим алата из [76] користе SMT решавач *Z3* за потребе доказивања. Алат описан у [131, 94] анализира сачуване процедуре (енгл. *stored procedures*) у језику PL/pgSQL.

Сви ови алати не разматрају еквиваленцију два фрагмента кода и усмерени су само на генерисање тест случајева. Аутор рада [197] разматра проблем верификације еквивалентности између пара програма који приступају бази података са различитим схемама (док се у овој тези подразумевају исте схеме и исти садржај) и фокусиран је на SQL трансакције. Систем *Blitz* [165] може проверавати еквивалентност SQL упита до на ограничену величину улаза и реализује синтетисање кода у контексту оптимизације упита.

Што се тиче језика SPARQL, он се најчешће користи из програмског језика *Java*, али за сада не постоје алати који анализирају овакву комбинацију програмских језика, тј. *Java* програме који садрже SPARQL упите, или на основу њих генеришу тест случајеве.

2.4.3 Еквивалентност програма и регресиона верификација

Одређивање еквиваленције два програма у случају када су ти програми слични зове се *регресиона верификација* (енгл. *regression verification*) [181]. Верификација еквиваленције подразумева формално доказивање одређених математичких својстава два програма. Постоје различите форме еквиваленције:

Парцијална еквиваленција (енгл. *partial equivalence*) два програма важи у случају да извршавања (која се заустављају) почевши од једнаких улаза дају једнаке излазе.

K-еквиваленција важи ако извршавања где петље[4] имају највише k итерација[5] почевши од једнаких улаза дају једнаке излазе.

Парцијална еквиваленција је неодлучиви проблем, док је k-еквиваленција (за неко одређено k, претпостављајући да се користе коначни домени променљивих) одлучива. При доказивању k-еквиваленције, избор праве вредности за k врло је важан. Велике вредности дају већи степен поверења, али повећавање вредности k може довести до проблема скалабилности (енгл. *scalability issues*). Неки алати заснивају се на претпоставци да се много грешака може открити у само једној итерацији [12, 80].

На тему доказивања функционалне еквивалентности програма који приступају базама података помоћу упита у неком упитном језику не постоје одговарајућа истраживања, али постоје слични релевантни проблеми, као што је појединачно разматрање еквивалентности упита или еквивалентности императивних програма.

Функционална еквивалентност императивног кода

Провера еквиваленције два императивна програма је неодлучив проблем и теоријски је разматран још у шездесетим годинама прошлог века [105]. Међутим, све до недавно, напредак је био ограничен и недовољно ефикасан за практичну употребу.

Развој метода регресионе верификације најчешће се води њиховим применама у различитим областима, као што су безбедност [3, 164], криптографија [153, 18], мултимедијални системи [190], компатибилност уназад и рефакторисање [202] и дизајн хардвера [107]. Ове технике нису потпуно аутоматизоване и захтевају додатно експертско знање.

Такође, постоје различити аутоматски методи регресионе верификације [95, 77, 13, 98], али су они обично ограничени у областима примене. На пример, за доказивање функционалне еквивалентности кода који садржи петље, у [95], петље се трансформишу у еквивалентан рекурзивни кôд, рекурзивни позиви се мењају неинтерпретираним функцијама, и на крају се покушава доказивање еквиваленције користећи предефинисане инваријанте. Пуна функционална еквивалентност се обично не разматра на великим програмима, и њихово понашање се само проверава k-еквиваленцијом, тј. петље се размотавају само до одређене границе [13]. У таквим ситуацијама, алати за проверу модела [193, 53, 136, 61, 186] имају своју улогу [192, 194].

У приступу предложеном у овој тези, ако кôд после упита није промењен, у неким случајевима може се доказати пуна еквиваленција чак и у присуству петљи, како је објашњено у поглављу 4.3. У осталим случајевима, разматра се само k-еквивалентност. Иако је она слабија форма еквиваленције, ипак пружа више поузданости у еквивалентност кода од тестирања, јер симболичко израчунавање може заменити велики број тестова и

[4]Ова врста еквиваленције се разматра само у случају императивног кода.

[5]У присуству рекурзивних функција, највише k рекурзивних позива је могуће.

чак при коришћењу малих вредности параметра k могу се открити неочекивани контра примери [165]. Од горе наведених, најсличнији приступу из ове тезе је приступ приказан у [95] јер и он моделује апстрактне делове кода неинтерпретираним функцијама, а резоновање се оставља SMT решавачима у теорији неинтерпретираних функција. У предложеном приспуту, за неке функције уводе се додатне аксиоме које моделују нека њихова својства.

Еквивалентност упита

Упити су еквивалентни ако су скупови одговора та два упита над било којом базом једнака. Проблем еквивалентности два упита на било ком упитном језику своди се на проверу релације садржаности упита у оба смера (поглавље 2.3). Наиме, скупови су једнаки, ако је један подскуп другог и обратно, што значи да је релација садржаности задовољена у оба смера, тј. први упит је садржан у другом и обратно.

Дакле, за проверу еквивалентности могу се користити решавачи баш тог проблема, рецимо за језик SQL представљени у поглављу 2.3.2, или решавачи проблема садржаности, који су за језик SPARQL представљени у поглављу 2.3.4.

Глава 3

Проблем садржаности упита у језику SPARQL

До сада је развијено више различитих решавача проблема садржаности SPARQL упита [42, 92, 125, 180], али и даље постоје многи нерешени проблеми, као што су недовољна покривеност различитих језичких конструката, ефикасност, разматрање RDF схеме (енгл. RDF SCHEMA *entailment regime*), и други. За сваки решавач проблема садржаности, врло је важно разматрати његову поузданост, тј. његов логички статус сагласности (енгл. *soundness*) и потпуности (енгл. *completeness*). Несагласни приступи су потпуно неприхватљиви, док је присутност потпуности уз сагласност врло пожељна карактеристика, могућа само за неке фрагменте језика.

У овој глави, представљена је оригинална процедура за решавање проблема садржаности упита у језику SPARQL свођењем на проблем задовољивости формуле у логици првог реда (FOL). Такође, представљена је имплементација те процедуре, у виду алата названог SpeCS (*SParql quEry Containment Solver*). Иницијалне идеје овог свођења и кратка евалуација дате су у [172]. Као машинерија за резоновање у алату, користе се ефикасни решавачи проблема испитивања задовољивости формула логике првог реда у односу на једну или више теорија, тј. SMT решавачи [15].

Предложен приступ покрива основни фрагмент језика SPARQL. Доказана је и сагласност и потпуност предложене процедуре за овај фрагмент. Све дефиниције дате су рекурзивно, а докази лема и теореми индуктивно, па се приступ може инкрементално проширити додатним језичким конструкцијама. Представљена је и темељна експериментална евалуација предложеног приступа. Поређење са постојећим савременим решавачима проблема садржаности упита у језику SPARQL на најзначајнијим скуповима примера за тестирање показало је да предложени приступ има боље перфомансе и већу покривеност подржаних језичких конструкција. Примери за тестирање, са свим паровима упита и њиховим спецификацијама, сви решавачи и њихови Јава омотачи (енгл. *wrappers*), заједно са табелама које садрже резултате представљене евалуације јавно су доступни на вебу [178].

Преглед главе: Поглавље 3.1 у кратким цртама даје увод у језик SPARQL, прецизно дефинише SPARQL семантку и проблем садржаности упита у језику SPARQL. Поглавље 3.2 доноси оригинално моделовање основног подскупа SPARQL упита и моделовање релације садржаности над конјунктивним упитима. Поглавља 3.3, 3.4 и 3.5 проширују покривен подскуп језика граф обрасцима који садрже подупите, унију и опционе обрасце, редом. Поглавље 3.6 проширује предложени приступ подржавајући SPARQL

негацију, приступ именованим графовима, шири подскуп израза и услова, као и разматрање релације садржаности узимајући у обзир RDF схему. Поглавље 3.7 даје преглед предложеног приступа у контексту релације стапања, која је слабија форма релације садржаности. Поглавље 3.8 представља имплементацију предложеног приступа и његову евалуацију, поредећи га са свим доступним савременим решавачима истог проблема на постојећим скуповима примера за тестирање.

3.1 Језик SPARQL и проблем садржаности упита

W3C (енгл. *World Wide Web Consortium*) дефинисао је граматику језика SPARQL у EBNF нотацији [155]. Њен упрошћени подскуп који се разматра надаље приказан је на слици 3.1. Објекти (инстанце) над којима се постављају SPARQL упити представљени су у форми интернационалних идентификатора ресурса (енгл. *Internationalized Resource Identifier* — IRI) који представљају уопштење јединствених идентификатора ресурса (енгл. *Uniform Resource Identifier* — URI) [74].

```
Query    ::= select Vars
             (from iri)*
             (from named iri)*
             where QPattern
Vars     ::= '*' | var+
QPattern::= '{' GPattern '}'
GPattern::= TPattern
          | GPattern '.' GPattern
          | GPattern filter Cond
          | '{' GPattern '}'
TPattern::= Subject Predicate Object
```

```
Subject  ::= var | iri | blankNode
Predicate::= var | iri
Object   ::= var | iri | blankNode
           | rdfLiteral
Expr     ::= var | iri | rdfLiteral
Cond     ::= Expr '=' Expr
           | UnOp Cond
           | Cond BinOp Cond
           | '(' Cond ')'
UnOp     ::= '!'
BinOp    ::= '&&' | '||'
```

Слика 3.1: Упрошћени подскуп граматике SPARQL упита

Пример 3.1 Примери упита који припадају језику ове граматике приказани су на слици 3.2. Свако решење упита Q_1 је у исто време и решење упита Q_2. Упит Q_1 тражи студенте основних студија (`?x a :UndergradStudent`), тј. све инстанце `?x` који припадају (предикат `a`) класи `:UndergradStudent`. Од свих њих, захтева се да похађају (предикат `:takesCourse`) неки курс `?y`, а њихово име (предикат `:name`) и број телефона (предикат `:phone`) су редом `?n` и `?tel`. Дакле, оператор `.` означава конјункцију захтева који су задати у форми уређених тројки, тј. субјекат-предикат-објекат. Оператор `filter` од свих таквих студената и њихових курсева које похађају (представљених идентификатором студента `?x`, курса `?y`, именом `?n` и бројем телефона `?tel` студента) издваја само оне чије је име Тим (`filter (?n = "Tim")`). На самом крају извршавања упита, оператор пројекције (`select ?x ?n ?y`), издваја такве студенте `?x`, њихова имена `?n` и похађане курсеве `?y`. Упит Q_2 тумачи се на сличан начин.

У тексту који следи, `V`, `I`, `B` и `L` означавају дате, међусобно дисјунктне, пребројиве скупове: `V` је скуп који садржи променљиве, `I` је скуп који садржи IRI-јеве, `B` је скуп

Упит Q_1

```
select ?x ?n ?y
where {
  ?x a              :UndergradStudent.
  ?x :takesCourse ?y .
  ?x :name          ?n .
  ?x :phone         ?tel
  filter (?n = "Tim")
}
```

Упит Q_2

```
select ?x ?n ?y
where {
  ?x a              :UndergradStudent.
  ?x :takesCourse ?y .
  ?x :name          ?n
}
```

Слика 3.2: Упит Q_1 (лево) је под-упит упита Q_2 (десно)

неименованих чворова (енгл. *blank nodes*) којима се означавају неименовани ресурси, а L је скуп литерала. Елементи ових скупова одговарају RDF термовима, тј. var, iri, blankNode и rdfLiteral у граматици са слике 3.1. Као у [42], било која унија ових скупова скраћено се означава навођењем свих елемената уније, на пример IBL := IUBUL.

RDF *тройлет* је елемент скупа IB × I × IBL. *Тройлет образац* дефинисан је нетерминалом TPattern у граматици. Може се приметити да је сваки триплет образац елемент скупа VIB × VI × VIBL, што значи да је сваки RDF триплет уједно и триплет образац по својој дефиницији. Елементи триплет обрасца називају се *субјекат*, *предикат* и *објекат*, као и одговарајући нетерминали. RDF триплет може се тумачити и као граф, где су субјекат и објекат чворови графа, а грана између њих означена је предикатом. RDF *граф* је граф који се састоји од RDF триплета који деле заједничке чворове. *Граф обрасци* и *упит обрасци* дефинисани су као нетерминали GPattern и QPattern у граматици. Граф обрасци могу садржати различите операторе, као што су ., filter и заграде[1]. Различити услови (описани нетерминалом Cond у граматици) који садрже изразе (описане нетерминалом Expr у граматици) могу бити део filter клаузуле.

Дефиниција 3.1 (RDF скуп података) RDF скуп података D је скуп

$$\{G_d, \langle i_1, G_1\rangle, ..., \langle i_n, G_n\rangle\},$$

где

- сваки G_k је RDF граф,
- скупови неименованих чворова у различитим графовима су дисјунктни, и
- сваки i_k је различит IRI.

Скуп података мора садржати граф G_d, који се назива *подразумевани граф* (енгл. *default graph*) скупа података D. Он дефинише функцију *df* са $df(D) := G_d$, која пресликава скуп података D у његов подразумевани граф. Скуп података може садржати нула или више парова $\langle i_k, G_k\rangle$, који се називају *именовани графови* (енгл. *named graphs*). Они дефинишу

[1] Други типови граф образаца покривени су у поглављима 3.3, 3.4, 3.5 и 3.6.

функцију $names$ са $names(\mathtt{D}) := \{\mathtt{i}_1, ..., \mathtt{i}_n\}$, која пресликава скуп података D у скуп IRI-ја његових именованих графова. Именовани графови такође дефинишу функцију gr, која има два аргумента. Први је RDF скуп података D, који се ради читљивости означава у индексу, и који означава скуп података који садржи именовани граф. Други аргумент је IRI, па се функција дефинише са:

$$gr_{\mathtt{D}}(\mathtt{i}_k) := \begin{cases} \mathtt{G}_k, & \text{ако } \langle \mathtt{i}_k, \mathtt{G}_k \rangle \in \mathtt{D} \\ \varnothing, & \text{у супротном} \end{cases}$$

RDF *обједињавање* подскупа $\{\mathtt{G}_{k_1}, ..., \mathtt{G}_{k_m}\}$ RDF графова[2] из скупа података D је граф $\mathtt{G} := \bigcup_{i=1}^{m} \mathtt{G}_{k_i}$, у ознаци $merge_{\mathtt{D}}(\mathtt{G}_{k_1}, ..., \mathtt{G}_{k_m})$. Слично функцији gr, $merge$ има аргумент D, записан у индексу, који означава RDF скуп података D, а који садржи RDF графове $\mathtt{G}_{k_1}, ..., \mathtt{G}_{k_m}$.

SPARQL упит извршава се над RDF скупом података који ће бити означаван са $\mathbb{D}$. Упит може прецизирати скуп података који ће се користити, *упитни скуп података*, а D ће бити његова ознака.

Дефиниција 3.2 (Упитни скуп података D) Упитни скуп података D који је одређен `from` и `from named` клаузулама упита над RDF скупом података $\mathbb{D}$ дефинише се на следећи начин:

- Ако упит не садржи ни `from` ни `from named` клаузуле, тада је $\mathtt{D} := \mathbb{D}$.
- У супротном, подразумевани граф упитног скупа података D дефинише се на следећи начин:
  - Ако упит садржи бар једну `from` клаузулу,

    $$df(\mathtt{D}) := merge_{\mathbb{D}}(\mathtt{G}_{k_1}, ..., \mathtt{G}_{k_m}),$$

    где IRI-јеви $\mathtt{i}_{k_j}, j \in \{1, \cdots, m\}$ референсани у `from` клаузулама дефинишу графове $\mathtt{G}_{k_j}$ у RDF обједињавању, тј. важи $\mathtt{G}_{k_j} := gr_{\mathbb{D}}(\mathtt{i}_{k_j}), j \in \{1, \cdots, m\}$.
  - Ако упит садржи `from named`, али не и `from` клаузуле, $df(\mathtt{D}) := \mathtt{G}_{\varnothing}$, тј. подразумевани граф упитног скупа података D је празан граф.

  Ако упит садржи `from` али не и `from named` клаузуле, упитни скуп података D ће садржати подразумевани и нула именованих графова. Иначе, именовани графови $\langle \mathtt{i}_{l_j}, \mathtt{G}_{l_j} \rangle$ упитног скупа података D дефинишу се IRI-јима $\mathtt{i}_{l_j}, j \in \{1, \cdots, n\}$ који се налазе у `from named` клаузулама, тј. сваки $\mathtt{G}_{l_j}$ у именованим графовима дефинише се са $\mathtt{G}_{l_j} := gr_{\mathbb{D}}(\mathtt{i}_{l_j}), j \in \{1, \cdots, n\}$.

Може се приметити да ако неко $\mathtt{i}_{k_j}$ из `from` клаузуле не припада скупу $names(\mathbb{D})$, по дефиницији 3.1, важи $gr_{\mathbb{D}}(\mathtt{i}_{k_j}) = \varnothing$.

Сваки триплет образац упита тражи се у RDF графу из упитног скупа података, који се тада назива *активни граф* (енгл. *active graph*). У упрошћеној граматици SPARQL упита приказаној на слици 3.1, активни граф је увек једнак подразумеваном графу упитног скупа података. Међутим, када граматика буде проширена `graph` клаузулом (у поглављу 3.6), активни граф може бити једнак и неком од именованих графова.

[2]Претпоставља се дисјунктност скупова неименованих чворова из различитих графова једног скупа података, због чега у функцији $merge_{\mathtt{D}}$ фигурише D.

Пример 3.2 Један активни граф који садржи податке о студентима, и над којим има смисла извршавати упите са слике 3.2 дат је на слици 3.3. Клаузуле `prefix` су део синтаксе и служе само компактнијем запису података. У графу постоје три студента `:stud001`, `:stud002` и `:stud003`, са именима `Tim`, `Berners`, `Lee`, редом, при чему први од њих има и податак о броју телефона. Прва два студента су студенти основних студија (класа `UndergradStudent`), док је трећи студент мастер студија (класа `MasterStudent`). Први студент похађа два курса `:course04` и `:course10`, чија су имена `Programming 1` и `Linked Data` и носе 9 и 5 ESBP поена, редом. Други такође похађа курс `:course10`, док трећи слуша курс чије је име `Semantic Web` који вреди 8 ESBP поена. Термови `:stud001`, `:stud002`, `:stud003`, `UndergradStudent`, `MasterStudent`, `:course04`, `:course10` из овог графа припадају скупу `I`, термови `"Tim"`, `"Berners"`, `"Lee"`, `"064/123-4-567"`, `"Programming 1"`, `"Linked Data"`, `"Semantic Web"`, `"9"^^xsd:integer`, `"8"^^xsd:integer` и `"7"^^xsd:integer` припадају скупу `L`, док терм `_:b1` представља неименовани чвор, тј. припада скупу `B`.

Неименовани чворови у RDF графовима у оквиру скупа података и неименовани чворови у граф обрасцима имају потпуно другачију семантику. Првопоменути представљају неименовани ресурс, тј. онај за који није познат IRI или литерал, али неименовани чвор указује да такав чвор постоји. Са друге стране, неименовани чворови у граф обрасцима третирају се као променљиве, а не као референце одређених неименованих чворова у скупу података. У даљем тексту, свако третирање неименованих чворова као променљивих биће јасно наглашено.

Интуитивно, резултат извшавања упита треба да повеже променљиве из упит обрасца са вредностима у активном графу. Такође, да би се упит евалуирао, потребно је повезати и неименоване чворове из упит обрасца са вредностима из активног графа, али ове вредности не могу ући у резултат упита. За скуп променљивих (и неименованих чворова) упита, може постојати више од једног скупа вредности које задовољавају услов наметнут упитом. Зато, резултат извршавања упита је скуп пресликавања која повезују скуп променљивих `V` и скуп `IBL`.

Пример 3.3 Резултат упита Q_1 са слике 3.2 повезује променљиве `?x`, `?n` и `?y` са одговарајућим термовима из `IBL`, који представљају одговарајуће студенте, њихова имена и похађане курсеве. Ако се упит извршава над скупом података $\mathbb{D}$ чији је подразумевани граф приказан на слици 3.3, једно такво повезивање је представљено пресликавањем чији је домен `{?x, ?n, ?y}` и које их пресликава на следећи начин:

$$\texttt{?x} \mapsto \texttt{:stud001}, \quad \texttt{?n} \mapsto \texttt{"Tim"}, \quad \texttt{?y} \mapsto \texttt{:course04}$$

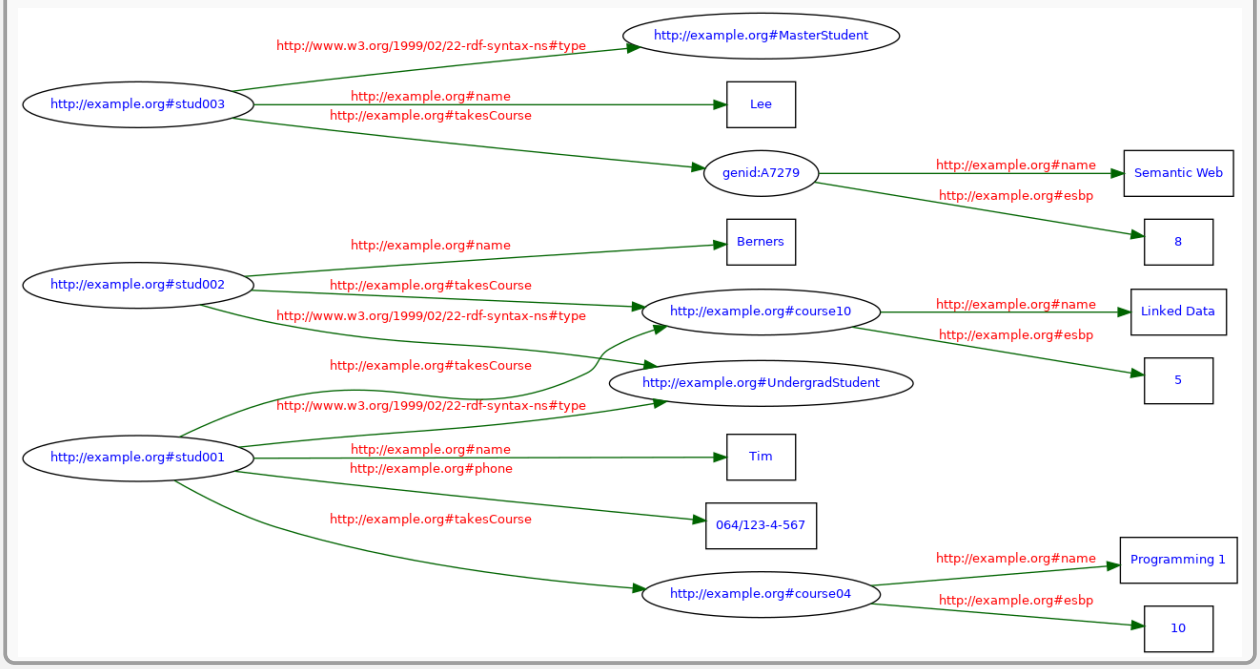
Такође, повезивање може бити и пресликавање истог домена, за које важи:

$$\texttt{?x} \mapsto \texttt{:stud001}, \quad \texttt{?n} \mapsto \texttt{"Tim"}, \quad \texttt{?y} \mapsto \texttt{:course10}$$

Дакле, резултат упита Q_1 је скуп поменутих пресликавања:

$$\left\{ \begin{array}{l} \{ \ \texttt{?x} \mapsto \texttt{:stud001}, \ \texttt{?n} \mapsto \texttt{"Tim"}, \ \texttt{?y} \mapsto \texttt{:course04} \ \}, \\ \{ \ \texttt{?x} \mapsto \texttt{:stud001}, \ \texttt{?n} \mapsto \texttt{"Tim"}, \ \texttt{?y} \mapsto \texttt{:course10} \ \} \end{array} \right\}$$

```
prefix     : <http://example.org#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

:stud001 rdf:type      :UndergradStudent .
:stud001 :name         "Tim" .
:stud001 :phone        "064/123-4-567" .
:stud001 :takesCourse  :course04 .
:stud001 :takesCourse  :course10 .

:stud002 rdf:type      :UndergradStudent .
:stud002 :name         "Berners" .
:stud002 :takesCourse  :course10 .

:stud003 rdf:type      :MasterStudent .
:stud003 :name         "Lee" .
:stud003 :takesCourse  _:b1 .

:course04 :name "Programming 1" .
:course04 :esbp "9"^^xsd:integer .

:course10 :name "Linked Data" .
:course10 :esbp "5"^^xsd:integer .

_:b1      :name "Semantic Web" .
_:b1      :esbp "8"^^xsd:integer .
```



Слика 3.3: Пример RDF графа са подацима о студентима и курсевима (горе) и његова графичка репрезентација (доле)

Слично, резултат упита Q_2 са слике 3.2 над истим подацима је

$$\{ \{ ?x \mapsto :stud001, ?n \mapsto "Tim", ?y \mapsto :course04 \}, \{ ?x \mapsto :stud001, ?n \mapsto "Tim", ?y \mapsto :course10 \}, \{ ?x \mapsto :stud002, ?n \mapsto "Berners", ?y \mapsto :course10 \} \}$$

Може се приметити да је сваки резултат упита $\mathtt{Q}_1$ (прва два пресликавања из примера) у исто време и резултат упита $\mathtt{Q}_2$. Како ће ово важити за сваки скуп података $\mathbb{D}$, упит $\mathtt{Q}_1$ је под-упит упита $\mathtt{Q}_2$.

Како су резултати извршавања упита скупови пресликавања, потребно је дефинисати различите операције над њима, што је и урађено дефиницијама које следе. У њима, $dom(m)$ означава домен пресликавања m.

Дефиниција 3.3 (Компатибилна пресликавања) Пресликавања m_1 и m_2 су *компатибилна*, у ознаци $m_1 \simeq m_2$, ако за свако x такво да $x \in dom(m_1) \cap dom(m_2)$ важи $m_1(x) = m_2(x)$.

Свака два пресликавања са дисјунктним доменима су компатибилна. За пресликавања m_1 и m_2 за које важи $m_1 \simeq m_2$, њихова унија, у ознаци $m_1 \cup m_2$, је добро дефинисано пресликавање, за које важи:

$$dom(m_1 \cup m_2) = dom(m_1) \cup dom(m_2)$$

$$(m_1 \cup m_2)(x) = \begin{cases} m_1(x), & x \in dom(m_1) \\ m_2(x), & \text{иначе} \end{cases}$$

Пресликавање $m_1 \cup m_2$ је компатибилно и са пресликавањем m_1 и са пресликавањем m_2, тј. важи $m_1 \cup m_2 \simeq m_1$ и $m_1 \cup m_2 \simeq m_2$.

Дефиниција 3.4 (Оператори над скуповима пресликавања) Нека су Ω_1 и Ω_2 скупови пресликавања. Оператори *уније*, *спајања*, *разлике* и *левог спољашњег спајања* дефинишу се са [149]:

$$\begin{aligned} \Omega_1 \cup \Omega_2 &:= \{m \mid m \in \Omega_1 \text{ или } m \in \Omega_2\} \\ \Omega_1 \bowtie \Omega_2 &:= \{m_1 \cup m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2, m_1 \simeq m_2\} \\ \Omega_1 \setminus \Omega_2 &:= \{m_1 \mid m_1 \in \Omega_1 \text{ и за свако } m_2 \in \Omega_2, \text{ не важи } m_1 \simeq m_2\} \\ \Omega_1 \mathbin{⟕} \Omega_2 &:= (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2) \end{aligned}$$

Пример 3.4 Нека су `?x`, `?y` и `?n` променљиве, а `:stud001`, `:stud002`, `:course04`, `:course10`, `"Tim"` и `"Berners"` елементи скупа `IBL` (као у примеру 3.3). Нека су Ω_1 и Ω_2 следећи скупови пресликавања:

$$\begin{aligned} \Omega_1 = \{ & \\ & \{ \ \texttt{?x} \mapsto \texttt{:stud001}, \ \texttt{?n} \mapsto \texttt{"Tim"} \ \}, \\ & \{ \ \texttt{?x} \mapsto \texttt{:stud002}, \ \texttt{?n} \mapsto \texttt{"Berners"} \ \} \\ \} & \\ \Omega_2 = \{ & \\ & \{ \ \texttt{?x} \mapsto \texttt{:stud001}, \ \texttt{?y} \mapsto \texttt{:course04} \ \}, \\ & \{ \ \texttt{?x} \mapsto \texttt{:stud001}, \ \texttt{?y} \mapsto \texttt{:course10} \ \} \\ \} & \end{aligned}$$

Тада су њихова унија, спајање, разлика и лево спољашње спајање редом следећа пресликавања:

$$
\begin{aligned}
\Omega_1 \cup \Omega_2 = \{ & \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?n} \mapsto \texttt{"Tim"} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud002}, \texttt{?n} \mapsto \texttt{"Berners"} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?y} \mapsto \texttt{:course04} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?y} \mapsto \texttt{:course10} \} \\
\} & \\
\Omega_1 \bowtie \Omega_2 = \{ & \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?n} \mapsto \texttt{"Tim"}, \texttt{?y} \mapsto \texttt{:course04} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?n} \mapsto \texttt{"Tim"}, \texttt{?y} \mapsto \texttt{:course10} \} \\
\} & \\
\Omega_1 \setminus \Omega_2 = \{ & \\
& \{ \texttt{?x} \mapsto \texttt{:stud002}, \texttt{?n} \mapsto \texttt{"Berners"} \} \\
\} & \\
\Omega_1 \ltimes \Omega_2 = \{ & \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?n} \mapsto \texttt{"Tim"}, \texttt{?y} \mapsto \texttt{:course04} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?n} \mapsto \texttt{"Tim"}, \texttt{?y} \mapsto \texttt{:course10} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud002}, \texttt{?n} \mapsto \texttt{"Berners"} \} \\
\} &
\end{aligned}
$$

Дефиниција 3.5 (Проширење, рестрикција, пројекција) Пресликавање m_1 је *проширење* пресликавања m_2, у ознаци $m_1 \succeq m_2$, ако су пресликавања m_1 и m_2 компатибилна и $dom(m_1) \supseteq dom(m_2)$. Тада, пресликавање m_2 је *рестрикција* пресликавања m_1, у ознаци $m_2 \preceq m_1$. *Пројекција* пресликавања m на скуп $\overline{x}$, у ознаци $m_{\overline{x}}$, је пресликавање такво да важи $dom(m_{\overline{x}}) = dom(m) \cap \overline{x}$ и $m_{\overline{x}} \preceq m$.

Ако су скупови $dom(m)$ и $\overline{x}$ дисјунктни, $m_{\overline{x}}$ је *празно пресликавање*, тј. пресликавање чији је домен празан скуп.

Дефиниција 3.6 (Оператор пројекције) Нека је Ω скуп пресликавања, и $\overline{x}$ скуп променљивих. Тада, *оператор пројекције* над скуповима $\overline{x}$ и Ω, у ознаци $\Pi_{\overline{x}}(\Omega)$, дефинише се са:

$$\Pi_{\overline{x}}(\Omega) \ := \ \{m_{\overline{x}} \mid m \in \Omega\}$$

Пример 3.5 Нека је Ω следећи скуп пресликавања.

$$
\begin{aligned}
\Omega = \{ & \\
& \{ \texttt{?x} \mapsto \texttt{:stud001}, \texttt{?y} \mapsto \texttt{:course04} \}, \\
& \{ \texttt{?x} \mapsto \texttt{:stud002}, \texttt{?n} \mapsto \texttt{"Berners"} \} \\
\} &
\end{aligned}
$$

Нека је $\overline{x}$ скуп променљивих $\{?x, ?n, ?z\}$. Тада је оператор пројекције над скуповима $\overline{x}$ и Ω следећи скуп пресликавања:

$$\Pi_{\overline{x}}(\Omega) = \{ \quad \{ \; ?x \mapsto \; :stud001 \; \} \quad \{ \; ?x \mapsto \; :stud002, \; ?n \mapsto \; "Berners" \; \}, \; \}$$

У даљем тексту, функција μ означаваће парцијално пресликавање скупа променљивих и неименованих чворова VB у скуп IBL. Функција $\mu_{x \to c}$ означаваће пресликавање за које важи $dom(\mu_{x \to c}) := \{x\}$ и $\mu_{x \to c}(v) := c$. У наредним дефиницијама, прати се скуповна семантика SPARQL упита и евалуације граф образаца [42, 148, 149, 5].[3]

Дефиниција 3.7 (Функција var) Нека је t RDF терм, tp триплет образац, E, E_1, E_2 изрази (дефинисани у граматици нетерминалом Expr), R, R_1, R_2 услови (дефинисани у граматици нетерминалом Cond), и gp, gp_1, gp_2 граф обрасци. Променљиве које се појављују у RDF терму t, триплет обрасцу tp, изразу E, услову R и граф обрасцу gp, у ознаци var(t), var(tp), var(E), var(R) и var(gp) редом, су

$$\mathtt{var(t)} := \begin{cases} \{t\}, & t \in VB \\ \varnothing, & t \in IL \end{cases}$$

$$\mathtt{var(tp)} := \{ \mathtt{var(s)} \cup \mathtt{var(p)} \cup \mathtt{var(o)}, \mathtt{tp} \text{ je } s \;\; p \;\; o \text{ и } s \in VIB,\; p \in VI,\; o \in VIBL$$

$$\mathtt{var(E)} := \{ \mathtt{var(t)}, \quad E \text{ je } t \text{ и } t \in VIL$$

$$\mathtt{var(R)} := \begin{cases} \mathtt{var}(E_1) \cup \mathtt{var}(E_2), & R \text{ je } E_1 = E_2 \\ \mathtt{var}(R_1), & R \text{ je } !R_1 \text{ или } R \text{ je } (R_1) \\ \mathtt{var}(R_1) \cup \mathtt{var}(R_2), & R \text{ je } R_1 \&\& R_2 \text{ или } R \text{ je } R_1 \,||\, R_2 \end{cases}$$

$$\mathtt{var(gp)} := \begin{cases} \mathtt{var(tp)}, & gp \text{ je } tp \\ \mathtt{var}(gp_1), & gp \text{ je } gp_1 \text{ filter } R \text{ или } gp \text{ je } \{gp_1\} \\ \mathtt{var}(gp_1) \cup \mathtt{var}(gp_2), & gp \text{ je } gp_1 \,.\, gp_2 \end{cases}$$

У тези се разматрају само граф обрасци $gp = gp_1$ filter R за које важи

$$\mathtt{var(R)} \subseteq \mathtt{var}(gp_1),$$

јер супротно није рачунски пожељно [148].

Пример 3.6 Променљиве које се појављују у упит обрасцу gp_1 упита Q_1 са слике 3.2 (лево) су:

$$\mathtt{var}(gp_1) = \{?x, ?y, ?n, ?tel\}$$

Скуп IBLe је проширење скупа IBL који садржи и додатну константу, именовану err, тј. IBLe := IBL $\cup \{err\}$. Интуитивно, ова се константа користи да означи стање грешке која се може појавити при евалуацији израза која није могућа у неком специфичном контексту. Према томе, то је вредност проширења $[[\cdot]]$ пресликавања μ на променљиве и неименоване чворове ван $dom(\mu)$, како је наведено у следећој дефиницији.

[3] Постоје и други типови семантика, на пример мултискуповна семантика [5, 6, 149]. Упоредна анализа [149] скуповне и мултискуповне семантике презентује разлоге зашто је скуповна семантика од основног значаја у развоју и имплементацији упитног језика.

Дефиниција 3.8 (Нотација $[[\cdot]]_\mu$) Вредност RDF терма t, израза E и триплет обрасца tp, у пресликавању μ, у ознаци $[[\mathtt{t}]]_\mu$, $[[\mathtt{E}]]_\mu$, и $[[\mathtt{tp}]]_\mu$ редом, је вредност из скупа IBLe, дефинисана са:

$$[[\mathtt{t}]]_\mu := \begin{cases} \mathtt{t}\,, & \mathtt{t} \in \mathtt{IL} \\ \mu(\mathtt{t})\,, & \mathtt{t} \in \mathtt{VB} \text{ и } \mathtt{t} \in dom(\mu) \\ \mathtt{err}\,, & \mathtt{t} \in \mathtt{VB} \text{ и } \mathtt{t} \notin dom(\mu) \end{cases}$$

$$[[\mathtt{E}]]_\mu := \Big\{ [[\mathtt{t}]]_\mu\,, \quad \mathtt{E} \text{ је } \mathtt{t},\ \mathtt{t} \in \mathtt{VIL}$$

$$[[\mathtt{tp}]]_\mu := \Big\{ [[\mathtt{s}]]_\mu\ [[\mathtt{p}]]_\mu\ [[\mathtt{o}]]_\mu\,, \quad \mathtt{tp} \text{ је } \mathtt{s}\ \mathtt{p}\ \mathtt{o} \text{ и } \mathtt{s} \in \mathtt{VIB},\ \mathtt{p} \in \mathtt{VI},\ \mathtt{o} \in \mathtt{VIBL}$$

Пример 3.7 Нека је пресликавање μ дефинисано са:

$$\mathtt{?x} \mapsto \mathtt{:stud001}, \quad \mathtt{?n} \mapsto \mathtt{"Tim"}, \quad \mathtt{?y} \mapsto \mathtt{:course04}$$

Тада је вредност триплет обрасца tp, где је $\mathtt{tp} = \mathtt{?x}\ \mathtt{:name}\ \mathtt{?n}$, следећа уређена тројка над скупом IBLe:

$$[[\mathtt{tp}]]_\mu = \mathtt{:stud001}\ \mathtt{:name}\ \mathtt{"Tim"}$$

Дефиниција 3.9 (Релација $\Vdash$) Нека су E, $\mathtt{E}_1$ и $\mathtt{E}_2$ изрази, R, $\mathtt{R}_1$ и $\mathtt{R}_2$ услови. Пресликавање μ задовољава услов R, у ознаци $\mu \Vdash \mathtt{R}$, ако:

$$\mu \Vdash \mathtt{R} := \begin{cases} [[\mathtt{E}_1]]_\mu \neq \mathtt{err},\ [[\mathtt{E}_2]]_\mu \neq \mathtt{err} \text{ и } [[\mathtt{E}_1]]_\mu = [[\mathtt{E}_2]]_\mu\,, & \mathtt{R} \text{ је } \mathtt{E}_1 = \mathtt{E}_2 \\ \text{не важи } \mu \Vdash \mathtt{R}_1\,, & \mathtt{R} \text{ је } \mathtt{!R}_1 \\ \mu \Vdash \mathtt{R}_1 \text{ и } \mu \Vdash \mathtt{R}_2\,, & \mathtt{R} \text{ је } \mathtt{R}_1\mathtt{\&\&R}_2 \\ \mu \Vdash \mathtt{R}_1 \text{ или } \mu \Vdash \mathtt{R}_2\,, & \mathtt{R} \text{ је } \mathtt{R}_1\mathtt{||R}_2 \\ \mu \Vdash \mathtt{R}_1\,, & \mathtt{R} \text{ је } \mathtt{(R}_1\mathtt{)} \end{cases}$$

Пример 3.8 Нека је μ пресликавање из примера 3.7. Тада μ задовољава R, где је R услов `?n = "Tim"`, тј. важи $\mu \Vdash \mathtt{R}$.

Дефиниција 3.10 (Евалуација граф обрасца) Нека је D RDF скуп података, G граф у оквиру скупа података D, tp триплет образац, gp, $\mathtt{gp}_1$, $\mathtt{gp}_2$ граф обрасци, и R услов. Евалуација граф обрасца над графом G у скупу података D, у ознаци $[\![\cdot]\!]^{\mathtt{D}}_{\mathtt{G}}$, дефинисана је рекурзивно са:

$$[\![\mathtt{tp}]\!]^{\mathtt{D}}_{\mathtt{G}} := \Big\{ \mu \mid dom(\mu) = \mathtt{var(tp)} \ \wedge\ [[\mathtt{tp}]]_\mu \in \mathtt{G} \Big\}$$

$$[\![\mathtt{gp}_1\ .\ \mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}} := [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} \bowtie [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$$

$$[\![\mathtt{gp\ filter\ R}]\!]^{\mathtt{D}}_{\mathtt{G}} := \Big\{ \mu \in [\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} \mid \mu \Vdash \mathtt{R} \Big\}$$

$$[\![\mathtt{\{gp\}}]\!]^{\mathtt{D}}_{\mathtt{G}} := [\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}}$$

Може се приметити да у претходној дефиницији, са тренутно подржаним граф обрасцима, скуп података D у ознаци $[\![\cdot]\!]^{\mathtt{D}}_{\mathtt{G}}$ нема сврху, али ће бити потребан када се уведу нови граф обрасци у поглављу 3.6.2.

Пример 3.9 Евалуација обрасца `tp` над графом `G` са слике 3.3, где је `tp` триплет образац `?x :name ?n` је следећи скуп пресликавања:

$$[\![\texttt{tp}]\!]^{\texttt{D}}_{\texttt{G}} = \{$$
$$\{ \quad \texttt{?x} \mapsto \quad \texttt{:stud001}, \quad \texttt{?n} \mapsto \quad \texttt{"Tim"} \quad \},$$
$$\{ \quad \texttt{?x} \mapsto \quad \texttt{:stud002}, \quad \texttt{?n} \mapsto \quad \texttt{"Berners"} \quad \},$$
$$\{ \quad \texttt{?x} \mapsto \quad \texttt{:stud003}, \quad \texttt{?n} \mapsto \quad \texttt{"Lee"} \quad \}$$
$$\}$$

Следећа лема повезује домен евалуације граф обрасца и променљивих у граф обрасцу.

Лема 3.1 *Нека је* `D` RDF *скуп података,* `G` *граф у оквиру скупа података* `D`*,* `gp` *граф образац, и* μ *пресликавање такво да* $\mu \in [\![gp]\!]^D_G$*. Тада важи:*

$$dom(\mu) = var(gp).$$

Доказ. Лема је доказана индукцијом по граф обрасцу `gp`.

`gp` је `tp`

По дефиницији 3.10, због $\mu \in [\![\texttt{tp}]\!]^{\texttt{D}}_{\texttt{G}}$, важи

$$dom(\mu) = \texttt{var(tp)}.$$

`gp` је $\texttt{gp}_1.\texttt{gp}_2$

По дефиницији 3.10, због $\mu \in [\![\texttt{gp}_1.\texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}$, важи

$$\mu \in [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}} \bowtie [\![\texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}.$$

По дефиницији 3.4, постоје пресликавања μ_1 и μ_2, таква да је $\mu_1 \simeq \mu_2$, $\mu = \mu_1 \cup \mu_2$ и

$$\mu_1 \in [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}} \text{ и } \mu_2 \in [\![\texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}.$$

Зато важи $dom(\mu) = dom(\mu_1) \cup dom(\mu_2)$. По индуктивној хипотези, важи

$$dom(\mu_1) = \texttt{var}(\texttt{gp}_1) \text{ и } dom(\mu_2) = \texttt{var}(\texttt{gp}_2).$$

Тада је $dom(\mu) = \texttt{var}(\texttt{gp}_1) \cup \texttt{var}(\texttt{gp}_2)$, тј. по дефиницији 3.7,

$$dom(\mu) = \texttt{var}(\texttt{gp}_1.\texttt{gp}_2).$$

`gp` је $\texttt{gp}_1$ `filter R`

По дефиницији 3.10, због $\mu \in [\![\texttt{gp}_1 \texttt{ filter R}]\!]^{\texttt{D}}_{\texttt{G}}$, важи

$$\mu \in [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}}.$$

По индуктивној хипотези, важи $dom(\mu) = \texttt{var}(\texttt{gp}_1)$. Тада, по дефиницији 3.7, важи и

$$dom(\mu) = \texttt{var}(\texttt{gp}_1 \texttt{ filter R}).$$

gp је $\{\mathtt{gp}_1\}$

По дефиницији 3.10, због $\mu \in [\![\{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$, важи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

По индуктивној хипотези, важи $dom(\mu) = \mathtt{var}(\mathtt{gp}_1)$. Тада, по дефиницији 3.7, важи и

$$dom(\mu) = \mathtt{var}(\{\mathtt{gp}_1\}).$$

□

Дефиниција 3.11 (Евалуација упита) *Евалуација* упита Q над скупом података $\mathbb{D}$, у ознаци $[\![\mathtt{Q}]\!]^{\mathbb{D}}$, дефинише се са

$$[\![\mathtt{Q}]\!]^{\mathbb{D}} \ := \ \Pi_{\overline{\mathtt{dv}}}([\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}),$$

где у оквиру упита Q, qpat је упит образац, $\overline{\mathtt{dv}}$ је скуп *пројектованих променљивих*, тј. променљивих које се налазе у select клаузули упита Q, и D је упитни скуп података.

Може се приметити да ако select клаузула не садржи пројекције, тј. упит Q је select∗ упит, скуп његових пројектованих променљивих једнак је скупу свих променљивих у његовом упит обрасцу, тј. var(qpat). Свако пресликавање из скупа $[\![\mathtt{Q}]\!]^{\mathbb{D}}$ назива се *пресликавање решења* (енгл. *solution mapping*) које одговара упиту Q над скупом података $\mathbb{D}$. Слично, свако пресликавање из скупа $[\![\mathtt{qpat}]\!]^{\mathtt{D}}_{\mathtt{G}}$ назива се *пресликавање обрасца* (енгл. *pattern instance mapping*) које одговара композицији RDF *пресликавања инстанци* (енгл. *instance mapping*), које пресликава неименоване чворове у RDF термове из скупа IBL, и пресликавања решења које променљивама из упита додељује такође RDF термове из скупа IBL. У случају да пресликавање μ из скупа $[\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}$ има дисјунктан домен са скупом $\overline{\mathtt{dv}}$, тада $[\![\mathtt{Q}]\!]^{\mathbb{D}}$ садржи празно пресликавање, које одговара празном SPARQL решењу. У случају да је скуп $[\![\mathtt{Q}]\!]^{\mathbb{D}}$ празан, тада не постоји резултат евалуације упита.

Дефиниција 3.12 (Релевантне променљиве $\overline{\mathtt{rv}}$) За RDF скуп података $\mathbb{D}$, упит Q и пресликавање μ такво да $\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$, све променљиве из $dom(\mu)$ називају се *релевантне променљиве* и означавају се $\overline{\mathtt{rv}}$.

Следећа лема повезује релевантне променљиве, променљиве из одговарајућег упит обрасца и пројектоване променљиве.

Лема 3.2 *Нека је* Q *упит,* qpat *његов упит образац,* $\overline{rv}$ *скуп релевантних променљивих и* $\overline{dv}$ *скуп његових пројектованих променљивих. Тада важи:*

$$\overline{rv} = var(qpat) \cap \overline{dv}.$$

Доказ. По дефиницији 3.12, постоји пресликавање μ и скуп података $\mathbb{D}$ такав да

$$\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}} \text{ и } \overline{\mathtt{rv}} = dom(\mu).$$

Тада, по дефиницији 3.11, важи

$$\mu \in \Pi_{\overline{\mathtt{dv}}}([\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}).$$

По дефиницији 3.6, постоји пресликавање μ' такво да је

$$\mu = \mu'_{\overline{\mathtt{dv}}} \text{ и } \mu' \in [\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По леми 3.1, важи $dom(\mu') = \mathtt{var}(\mathtt{qpat})$. Тада, по дефиницији 3.5, важи и

$$dom(\mu) = dom(\mu') \cap \overline{\mathtt{dv}} = \mathtt{var}(\mathtt{qpat}) \cap \overline{\mathtt{dv}},$$

тј.

$$\overline{\mathtt{rv}} = \mathtt{var}(\mathtt{qpat}) \cap \overline{\mathtt{dv}}.$$

□

Дефиниција 3.13 (Задовољивост и незадовољивост упита) Упит $\mathtt{Q}$ је *задовољив* ако постоји RDF скуп података $\mathbb{D}$ и пресликавање μ такво да $\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$. У супротном, упит $\mathtt{Q}$ је *незадовољив*.

Дефиниција 3.14 (Садржаност упита, над-упит, под-упит, еквивалентност упита) Нека су $\mathtt{Q}_1$ и $\mathtt{Q}_2$ упити. Упит $\mathtt{Q}_1$ је *садржан* у упиту $\mathtt{Q}_2$, у ознаци $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$, ако за сваки RDF скуп података $\mathbb{D}$ важи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$. Упит $\mathtt{Q}_2$ назива се *над-упит* (енгл. *super-query*), а упит $\mathtt{Q}_1$ *под-упит* (енгл. *sub-query*). Упит $\mathtt{Q}_1$ је *еквивалентан* упиту $\mathtt{Q}_2$, у ознаци $\mathtt{Q}_1 \equiv \mathtt{Q}_2$, ако за сваки RDF скуп података $\mathbb{D}$ важи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} = [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$.

Може се приметити да је незадовољив упит под-упит било ког другог упита.

Дефиниција 3.15 (Проблем садржаности упита) За дата два упита $\mathtt{Q}_1$ и $\mathtt{Q}_2$, *проблем садржаности упита* представља утврђивање да ли је упит $\mathtt{Q}_1$ садржан у упиту $\mathtt{Q}_2$, тј. да ли важи $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$.

3.2 Моделовање конјунктивних упита у језику SPARQL

Граматика дата на слици 3.1 садржи најчешће коришћене конструкте у језику SPARQL (конјунктивни упити са `filter` клаузулама), и упрошћена је зарад једноставности. У овом поглављу, описан је приступ моделовању SPARQL упита дефинисаних овом граматиком, тј. њихова трансформација у FOL формуле које ће се користити за резоновање о различитим односима између упита. Моделовање упита који садрже додатне језичке конструкције описано је у поглављима 3.3, 3.4, 3.5 и 3.6.

Дефиниција 3.16 (Сигнатура $\mathcal{L}$ која одговара упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$) Сигнатура $\mathcal{L}$ FOL теорије која се користи за моделовање односа између SPARQL упита $\mathtt{Q}_1$ и $\mathtt{Q}_2$ је уређена тројка $(\mathcal{F}_s, \mathcal{P}_s, ar)$, где је $\mathcal{F}_s$ скуп функцијских симбола, $\mathcal{P}_s$ скуп предикатских симбола, и ar функција арности. Прецизније:

- $\mathcal{F}_s := \mathcal{C} \cup \mathcal{F}$ је скуп функцијских сибмола, где:
  - $\mathcal{C}$ је скуп функцијских симбола арности 0, тј. скуп константи које одговарају литералима и IRI-јима који се појављују у упитима. Такође садржи и константу *err* која одговара `err` из скупа `IBLe`.
  - $\mathcal{F}$ је празан скуп. Касније ће се користити за скуп функцијских симбола који одговарају уграђеним функцијама у језику SPARQL (неке од њих обрађене су у поглављу 3.6, на пример *datatype*).
- $\mathcal{P}_s := \mathcal{P} \cup \{\beta_n, \beta_d\}$ је скуп предикатских симбола, где:
  - $\mathcal{P}$ је скуп који садржи предикатски симбол једнакости ($=$). Овај скуп може се прошитити да садржи и друге предикатске симболе који одговарају уграђеним функцијама у језику SPARQL које враћају тачно/нетачно (неке од њих обрађене су у поглављу 3.6, на пример *isliteral*).
  - β_d и β_n су предикатски симболи који интуитивно одговарају припадности RDF триплета подразумеваном графу из RDF скупа података $\mathbb{D}$, или именованом графу одређеним присутним IRI-јем, редом.
- Функција арности дефинисана је са:

$$ar(\alpha) := \begin{cases} 0\,, & \text{ако } \alpha \in \mathcal{C}, \\ 2\,, & \text{ако } \alpha \in \mathcal{P}, \\ 3\,, & \text{ако } \alpha \in \mathcal{P}_s \text{ и } \alpha \text{ је } \beta_d, \\ 4\,, & \text{ако } \alpha \in \mathcal{P}_s \text{ и } \alpha \text{ је } \beta_n. \end{cases}$$

Скуп променљивих, који интуитивно одговара променљивама и неименованим чворовима из SPARQL упита $\mathtt{Q}_1$ и $\mathtt{Q}_2$, означава се $\mathcal{V}$.

3.2.1 Трансформација упита у формуле

Упит описан граматиком датом на слици 3.1 састоји се од клаузула `select`, `from`, `from named` и `where`. У овом поглављу биће описан начин трансформације овог типа упита у FOL формулу Φ.

Дефиниција 3.17 (Функција $\sigma_{\mathtt{t}}$) Функција $\sigma_{\mathtt{t}}$ пресликава термове из SPARQL упита у одговарајуће променљиве и константе из описане сигнатуре, тј. функција $\sigma_{\mathtt{t}}$ је бијективна функција која пресликава елементе из скупа `VIBLe` у скуп $\mathcal{V} \cup \mathcal{C}$ са:

$$\sigma_{\mathtt{t}}(\mathtt{t}) \ := \begin{cases} c\ (c \in \mathcal{C}), & \text{ако је } \mathtt{t} = \mathtt{c} \text{ и } \mathtt{c} \in \mathtt{IL}, \\ x\ (x \in \mathcal{V}), & \text{ако је } \mathtt{t} = \mathtt{x} \text{ и } \mathtt{x} \in \mathtt{VB}, \\ err, & \text{ако је } \mathtt{t} = \mathtt{err}. \end{cases}$$

Ако се упит извршава над RDF скупом података $\mathbb{D}$, може се приметити да, пратећи дефиницију 3.2, подразумевани граф упитног скупа података `D` може бити

- једнак подразумеваном графу скупа података $\mathbb{D}$,
- RDF обједињавање једног или више графова, или
- празан граф $\mathtt{G}_{\varnothing}$.

Следећа дефиниција оправдава употребу нотације позива функције над скупом елемената њеног домена.

Дефиниција 3.18 (Функција над скупом) Нека је A скуп елемената, и f функција за коју важи $A \subseteq dom(f)$. Функција f над скупом A, у ознаци $f(A)$, означава скуп $\{f(x) \mid x \in A\}$.

Дефиниција 3.19 (Функција cx) Претпостављајући да се SPARQL упит извршава над RDF скупом података $\mathbb{D}$, и да је упитни скуп података `D`, функција cx пресликава активни граф `G` у скуп одговарајућих граф IRI константи на следећи начин:

- ако је активни граф `G` подразумевани граф упитног скупа података `D`, тада

$$cx(\mathtt{G}) \ := \begin{cases} \sigma_{\mathtt{t}}(\{\mathtt{i}_{k_1}, \cdots, \mathtt{i}_{k_m}\}), & \text{ако } \mathtt{G} = merge_{\mathbb{D}}(\mathtt{G}_{k_1}, ..., \mathtt{G}_{k_m}) \text{ и } gr_{\mathbb{D}}(\mathtt{i}_{k_j}) = \mathtt{G}_{k_j}, \\ & j \in \{1, \cdots, m\}, \\ \varnothing, & \text{ако } \mathtt{G} = \mathtt{G}_{\varnothing}, \end{cases}$$

- ако је активни граф `G` именовани граф упитног скупа података `D`, тада

$$cx(\mathtt{G}) \ := \ \sigma_{\mathtt{t}}(\{\mathtt{i}\}), \text{ где је } gr_{\mathbb{D}}(\mathtt{i}) = \mathtt{G}.$$

Може се приметити да функција cx није дефинисана на подразумеваном графу RDF скупа података $\mathbb{D}$, јер не постоји IRI придружен њему.

Следећа дефиниција уводи функцију σ за трансформисање SPARQL конструкта у одговарајуће FOL формуле из дате SPARQL сигнатуре. Пуна нотација функције укључује два аргумента: први је терм, израз, услов или образац, док је други активни RDF граф `G` који се користи за претрагу обрасцем у оквиру упита. Како трансформација термова, израза и услова не зависи од графа `G`, да би формуле биле једноставније, користи се функција σ само са једним аргументом (ово је случај и када је `G` очигледно из контекста). Када је неопходан, други аргумент је присутан у суперскрипту, тј. $\sigma^{\mathtt{G}}$.

Дефиниција 3.20 (Функција σ) Функција σ дефинише се рекурзивно у зависности од првог аргумента, тј. ако је аргумент:

- **Терм** `t`
  - ако $\texttt{t} \in \texttt{VIBL}$, тада:
    $\sigma(\texttt{t}) := \sigma_{\texttt{t}}(\texttt{t})$,
- **Израз** `E`
  - ако $\texttt{E} = \texttt{e}$ и $\texttt{e} \in \texttt{VILe}$, тада:
    $\sigma(\texttt{E}) := \sigma_{\texttt{t}}(\texttt{e})$,
- **Услов** `R`
  - ако је `R` оператор једнакости = који пореди изразе $\texttt{E}_1$ и $\texttt{E}_2$, резултат је одговарајући предикат =:
    $\sigma(\texttt{E}_1 = \texttt{E}_2) := \sigma(\texttt{E}_1) = \sigma(\texttt{E}_2)$,
  - ако је `R` логички оператор над условима $\texttt{R}_1$ и $\texttt{R}_2$, резултат је одговарајући логички везник у FOL-у:
    $\sigma(\texttt{! R}_1) := \neg\, \sigma(\texttt{R}_1)$,
    $\sigma(\texttt{R}_1 \texttt{ \&\& } \texttt{R}_2) := \sigma(\texttt{R}_1) \wedge \sigma(\texttt{R}_2)$,
    $\sigma(\texttt{R}_1 \texttt{ || } \texttt{R}_2) := \sigma(\texttt{R}_1) \vee \sigma(\texttt{R}_2)$,
  - ако је `R` услов $\texttt{R}_1$ у загради, резултат је одговарајућа формула:
    $\sigma(\texttt{(R}_1\texttt{)}) := \sigma(\texttt{R}_1)$,
- **Граф образац** `gp`, где се RDF граф `G` користи за проналажење пресликавања
  - ако је `gp` триплет образац `tp` у форми `s p o`, где $\texttt{s} \in \texttt{VIB}, \texttt{p} \in \texttt{VI}, \texttt{o} \in \texttt{VIBL}$, резултат је:

$$\sigma^{\texttt{G}}(\texttt{tp}) := \begin{cases} \beta_d(\sigma(\texttt{s}), \sigma(\texttt{p}), \sigma(\texttt{o}))\,, & \text{ако } \texttt{G} \notin dom(cx) \\ \bigvee\limits_{i_j \in cx(\texttt{G})} \beta_n(\sigma(\texttt{s}), \sigma(\texttt{p}), \sigma(\texttt{o}), i_j)\,, & \text{ако } \texttt{G} \in dom(cx) \text{ и } cx(\texttt{G}) \neq \varnothing \\ \bot\,, & cx(\texttt{G}) = \varnothing, \end{cases}$$

  - ако је `gp` конјункција два обрасца, $\texttt{gp}_1$ и $\texttt{gp}_2$, резултат је следећа конјункција:
    $\sigma^{\texttt{G}}(\texttt{gp}_1\texttt{ . gp}_2) := \sigma^{\texttt{G}}(\texttt{gp}_1) \wedge \sigma^{\texttt{G}}(\texttt{gp}_2)$,
  - ако је `gp` клаузула `filter` примењена на образац $\texttt{gp}_1$ са условом `R`, резултат је следећа конјункција:[4]
    $\sigma^{\texttt{G}}(\texttt{gp}_1\texttt{ filter R}) := \sigma^{\texttt{G}}(\texttt{gp}_1) \wedge \sigma(\texttt{R})$,
  - ако је `gp` граф образац $\texttt{gp}_1$ у витичастим заградама, резултат је примена функције σ на њега:
    $\sigma^{\texttt{G}}(\{\texttt{gp}_1\}) := \sigma^{\texttt{G}}(\texttt{gp}_1)$.

Може се приметити да контекст не може бити промењен у случају оператора `.`, `filter` и витичастих загради. Формула која одговара упиту датом на слици 3.2 приказана је на слици 3.4.

У овако изграђеним FOL формулама скуп њихових променљивих уводи се следећом дефиницијом.

[4] Оба оператора, и `.` и `filter`, моделују се оператором конјункције у FOL-у.

$$\begin{array}{llll} \beta_d(x_v, a_i, UndergradStudent_i) & \wedge & \beta_d(x_v, takesCourse_i, y_v) & \wedge \\ \beta_d(x_v, name_i, n_v) & \wedge & \beta_d(x_v, phone_i, tel_v) & \wedge \quad n_v = Tim_l \end{array}$$

Слика 3.4: Формула одговара упиту датом на слици 3.2 (лево). Ради читљивости, сваки IRI, литерал и променљива означени су одговарајућим индексом i, l и v, редом.

Дефиниција 3.21 (Функција *var* над формулама) Нека је Φ FOL формула. $var(\Phi)$ означава скуп свих слободних променљивих које се појављују у формули Φ.

Следеће четири леме повезују функције `var`, *var* и σ.

Лема 3.3 *Нека је* `t` RDF *терм. Тада важи:*

$$var(\sigma(\mathtt{t})) = \sigma(\mathtt{var}(\mathtt{t})).$$

Доказ. Једнакост се доказује индукцијом по терму `t`.

`t` је `c`, `c` ∈ `IL`

$$\begin{array}{lll} & var(\sigma(\mathtt{c})) & = \\ \text{(по дефиницији 3.20)} & & = var(\sigma_{\mathtt{t}}(\mathtt{c})) \\ \text{(по дефиницији 3.17)} & & = var(c) \\ \text{(по дефиницији 3.21)} & & = \varnothing \\ \text{(по дефиницији 3.18)} & & = \sigma(\varnothing) \\ \text{(по дефиницији 3.7)} & & = \sigma(\mathtt{var}(\mathtt{c})) \end{array}$$

`t` је `x`, `x` ∈ `VB`

$$\begin{array}{lll} & var(\sigma(\mathtt{x})) & = \\ \text{(по дефиницији 3.20)} & & = var(\sigma_{\mathtt{t}}(\mathtt{x})) \\ \text{(по дефиницији 3.17)} & & = var(x) \\ \text{(по дефиницији 3.21)} & & = \{x\} \\ \text{(по дефиницији 3.18)} & & = \sigma(\{\mathtt{x}\}) \\ \text{(по дефиницији 3.7)} & & = \sigma(\mathtt{var}(\mathtt{x})) \end{array}$$

□

Лема 3.4 *Нека је* `E` *израз. Тада важи:*

$$var(\sigma(\mathtt{E})) = \sigma(\mathtt{var}(\mathtt{E})).$$

Доказ. Једнакост се доказује индукцијом по изразу `E`.

E је t

$$\begin{aligned} var(\sigma(\texttt{t})) \quad &= \\ \text{(по леми 3.3)} \quad &= \sigma(\texttt{var(t)}) \end{aligned}$$

□

Лема 3.5 *Нека је* R *услов. Тада важи:*

$$var(\sigma(R)) = \sigma(var(R)).$$

Доказ. Једнакост се доказује индукцијом по услову R.

R је $\texttt{E}_1 = \texttt{E}_2$

$$\begin{aligned} var(\sigma(\texttt{E}_1 = \texttt{E}_2)) \quad &= \\ \text{(по дефиницији 3.20)} \quad &= var(\sigma(\texttt{E}_1) = \sigma(\texttt{E}_2)) \\ \text{(по дефиницији 3.21)} \quad &= var(\sigma(\texttt{E}_1)) \cup var(\sigma(\texttt{E}_2)) \\ \text{(по леми 3.4)} \quad &= \sigma(\texttt{var}(\texttt{E}_1)) \cup \sigma(\texttt{var}(\texttt{E}_2)) \\ \text{(по дефиницији 3.18)} \quad &= \sigma(\texttt{var}(\texttt{E}_1) \cup \texttt{var}(\texttt{E}_2)) \\ \text{(по дефиницији 3.7)} \quad &= \sigma(\texttt{var}(\texttt{E}_1 = \texttt{E}_2)) \end{aligned}$$

R је $\texttt{!R}_1$

$$\begin{aligned} var(\sigma(\texttt{!R}_1)) \quad &= \\ \text{(по дефиницији 3.20)} \quad &= var(\neg\sigma(\texttt{R}_1)) \\ \text{(по дефиницији 3.21)} \quad &= var(\sigma(\texttt{R}_1)) \\ \text{(по индуктивној хипотези)} \quad &= \sigma(\texttt{var}(\texttt{R}_1)) \\ \text{(по дефиницији 3.7)} \quad &= \sigma(\texttt{var}(\texttt{!R}_1)) \end{aligned}$$

R је $\texttt{R}_1\texttt{\&\&}\texttt{R}_2$

$$\begin{aligned} var(\sigma(\texttt{R}_1\texttt{\&\&}\texttt{R}_2)) \quad &= \\ \text{(по дефиницији 3.20)} \quad &= var(\sigma(\texttt{R}_1) \wedge \sigma(\texttt{R}_2)) \\ \text{(по дефиницији 3.21)} \quad &= var(\sigma(\texttt{R}_1)) \cup var(\sigma(\texttt{R}_2)) \\ \text{(по индуктивној хипотези)} \quad &= \sigma(\texttt{var}(\texttt{R}_1)) \cup \sigma(\texttt{var}(\texttt{R}_2)) \\ \text{(по дефиницији 3.18)} \quad &= \sigma(\texttt{var}(\texttt{R}_1) \cup \texttt{var}(\texttt{R}_2)) \\ \text{(по дефиницији 3.7)} \quad &= \sigma(\texttt{var}(\texttt{R}_1\texttt{\&\&}\texttt{R}_2)) \end{aligned}$$

R је $\texttt{R}_1\texttt{||}\texttt{R}_2$

Доказ је сличан претходном случају.

R је $\texttt{(R}_1\texttt{)}$

Доказ је сличан случају R је $\texttt{!R}_1$.

□

Лема 3.6 *Нека је gp граф образац. Ако су сви активни графови који се користе при претрази обрасца gp непразни, тада важи*

$$var(\sigma(gp)) = \sigma(\mathtt{var}(gp)).$$

Доказ. Једнакост се доказује индукцијом по граф обрасцу gp.

gp је tp, где је tp једнако s p o

Нека је G активни граф који се користи за при претрази обрасца tp.
Ако $\mathtt{G} \notin dom(cx)$:

$$\begin{aligned}
& var(\sigma^{\mathtt{G}}(\mathtt{s\ p\ o})) = \\
\text{(по дефиницији 3.20)} \quad & = var(\beta_d(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}))) \\
\text{(по дефиницији 3.21)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \mathcal{V} \\
\text{(по дефиницији 3.17)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \sigma_{\mathtt{t}}(\mathtt{VB}) \\
\text{(по дефиницији 3.20)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \sigma(\mathtt{VB}) \\
\text{(по дефиницији 3.18)} \quad & = \sigma(\{\mathtt{s}, \mathtt{p}, \mathtt{o}\} \cap \mathtt{VB}) \\
\text{(по дефиницији уније)} \quad & = \sigma((\{\mathtt{s}\} \cup \{\mathtt{p}\} \cup \{\mathtt{o}\}) \cap \mathtt{VB}) \\
\text{(по дистрибутивности)} \quad & = \sigma((\{\mathtt{s}\} \cap \mathtt{VB}) \ \cup (\{\mathtt{p}\} \cap \mathtt{VB}) \cup (\{\mathtt{o}\} \cap \mathtt{VB})) \\
\text{(по дефиницији 3.7)} \quad & = \sigma(\mathtt{var(s)} \cup \mathtt{var(p)} \cup \mathtt{var(o)})) \\
\text{(по дефиницији 3.7)} \quad & = \sigma(\mathtt{var(s\ p\ o)})
\end{aligned}$$

Ако $\mathtt{G} \in dom(cx)$, како је G непразан граф, по дефиницији 3.19, важи $cx(\mathtt{G}) \neq \varnothing$.

$$\begin{aligned}
& var(\sigma^{\mathtt{G}}(\mathtt{s\ p\ o})) = \\
\text{(по дефиницији 3.20)} \quad & = var(\bigvee_{i_j \in cx(\mathtt{G})} \beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i_j)) \\
\text{(по дефиницији 3.21)} \quad & = \bigcup_{i_j \in cx(\mathtt{G})} var(\beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i_j)) \\
\text{(по дефиницији 3.19)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \mathcal{V} \\
\text{(по дефиницији 3.17)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \sigma_{\mathtt{t}}(\mathtt{VB}) \\
\text{(по дефиницији 3.20)} \quad & = \{\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})\} \cap \sigma(\mathtt{VB}) \\
\text{(по дефиницији 3.18)} \quad & = \sigma(\{\mathtt{s}, \mathtt{p}, \mathtt{o}\} \cap \mathtt{VB}) \\
\text{(по дефиницији уније)} \quad & = \sigma((\{\mathtt{s}\} \cup \{\mathtt{p}\} \cup \{\mathtt{o}\}) \cap \mathtt{VB}) \\
\text{(по дистрибутивности)} \quad & = \sigma((\{\mathtt{s}\} \cap \mathtt{VB}) \ \cup (\{\mathtt{p}\} \cap \mathtt{VB}) \cup (\{\mathtt{o}\} \cap \mathtt{VB})) \\
\text{(по дефиницији 3.7)} \quad & = \sigma(\mathtt{var(s)} \cup \mathtt{var(p)} \cup \mathtt{var(o)})) \\
\text{(по дефиницији 3.7)} \quad & = \sigma(\mathtt{var(s\ p\ o)})
\end{aligned}$$

gp је $\mathtt{gp}_1.\mathtt{gp}_2$

$$\begin{aligned}
& var(\sigma(\mathtt{gp}_1.\mathtt{gp}_2)) = \\
\text{(по дефиницији 3.20)} \quad & = var(\sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{gp}_1)) \\
\text{(по дефиницији 3.21)} \quad & = var(\sigma(\mathtt{gp}_1)) \cup var(\sigma(\mathtt{gp}_1)) \\
\text{(по индуктивној хипотези)} \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1)) \cup \sigma(\mathtt{var}(\mathtt{gp}_1)) \\
\text{(по дефиницији 3.18)} \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1) \cup \mathtt{var}(\mathtt{gp}_2)) \\
\text{(по дефиницији 3.7)} \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1.\mathtt{gp}_2))
\end{aligned}$$

gp је $\mathtt{gp}_1$ `filter R`

$$
\begin{aligned}
var(\sigma(\mathtt{gp}_1 \texttt{ filter R})) &= \\
\text{(по дефиницији 3.20)} &= var(\sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{R})) \\
\text{(по дефиницији 3.21)} &= var(\sigma(\mathtt{gp}_1)) \cup var(\sigma(\mathtt{R})) \\
\text{(по индуктивној хипотези)} &= \sigma(\mathtt{var}(\mathtt{gp}_1)) \cup \mathtt{var}(\sigma(\mathtt{R})) \\
\text{(по леми 3.5)} &= \sigma(\mathtt{var}(\mathtt{gp}_1)) \cup \sigma(\mathtt{var}(\mathtt{R})) \\
\text{(по дефиницији 3.18)} &= \sigma(\mathtt{var}(\mathtt{gp}_1) \cup \mathtt{var}(\mathtt{R})) \\
\text{(како је } \mathtt{var}(\mathtt{R}) \subseteq \mathtt{var}(\mathtt{gp}_1)) &= \sigma(\mathtt{var}(\mathtt{gp}_1)) \\
\text{(по дефиницији 3.7)} &= \sigma(\mathtt{var}(\mathtt{gp}_1 \texttt{ filter R}))
\end{aligned}
$$

gp је $\{\mathtt{gp}_1\}$

$$
\begin{aligned}
var(\sigma(\{\mathtt{gp}_1\}) &= \\
\text{(по дефиницији 3.20)} &= var(\sigma(\mathtt{gp}_1)) \\
\text{(по индуктивној хипотези)} &= \sigma(\mathtt{var}(\mathtt{gp}_1)) \\
\text{(по дефиницији 3.7)} &= \sigma(\mathtt{var}(\{\mathtt{gp}_1\}))
\end{aligned}
$$

□

У тексту који следи, користи се нотација $\exists\overline{a}$ као скраћени запис $\exists a_1 ... \exists a_n$, као и $\forall\overline{a}$ уместо $\forall a_1 ... \forall a_n$, када је n познато из контекста.

Дефиниција 3.22 (Формула $\Phi(\overline{x})$ која одговара упиту Q) Нека је Q упит са упит обрасцем {qpat}, и D одговарајући упитни скуп података. Нека $\overline{x}$ означава скуп неких промењивих из $\mathcal{V}$, а $\overline{ov}$ означава све остале слободне промењиве које се појављују у формули $\sigma(\mathtt{qpat})$, тј. $\overline{ov} := var(\sigma(\mathtt{qpat})) \setminus \overline{x}$. Формула $\Phi(\overline{x})$ која одговара упиту Q дефинише се са:

$$\exists\overline{ov}\ \sigma^{df(\mathtt{D})}(\mathtt{qpat})$$

Ако $\overline{dv}$ означава промењиве из скупа $\mathcal{V}$ које одговарају пројектованим промењивама упита Q, тада, интуитивно, промењиве $\overline{ov}$ у формули $\Phi(\overline{dv})$ која одговара упиту Q означавају промењиве из скупа $\mathcal{V}$ које одговарају промењивама из скупа V које се користе у упиту, али нису издвојене у `select` клаузули.

Пример 3.10 На слици 3.5 дат је пример формуле Φ.

Лема 3.7 *Нека је $\overline{rv}$ скуп променљивих из $\mathcal{V}$ које одговарају релевантним променљивама упита Q и нека је $\Phi(\overline{rv})$ формула која одговара упиту Q. Тада важи:*

$$var(\Phi(\overline{rv})) = \overline{rv}.$$

$$\exists tel_v \Big(\beta_d(x_v, a_i, UndergradStudent_i) \wedge \beta_d(x_v, takesCourse_i, y_v) \wedge \beta_d(x_v, name_i, n_v) \wedge \beta_d(x_v, phone_i, tel_v) \wedge n_v = Tim_l\Big)$$

Слика 3.5: Формула $\Phi(\overline{dv})$ одговара упиту $\mathtt{Q}_1$ са слике 3.2 (лево). Ради читљивости, сваки IRI, литерал и променљива означени су одговарајућим индексом i, l и v, редом.

Доказ. Нека су qpat и D упит образац и упитни скуп података упита Q, редом. Нека $\overline{ov}$ означава $var(\sigma(\mathtt{qpat})) \setminus \overline{rv}$. Тада важи:

$$\begin{aligned}
& var(\Phi(\overline{rv})) = \\
\text{(по дефиницији 3.22)} \quad & = var(\exists \overline{ov}\ \sigma^{df(\mathtt{D})}(\mathtt{qpat})) \\
\text{(по дефиницији 3.21)} \quad & = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})) \setminus \overline{ov} \\
\text{(по дефиницији 3.22)} \quad & = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})) \setminus (var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})) \setminus \overline{rv}) \\
\text{(по дефиницији } \setminus \text{ и } \cap) \quad & = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})) \cap \overline{rv} \\
\text{(како је } \sigma(\overline{\mathtt{rv}}) = \overline{rv}) \quad & = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})) \cap \sigma(\overline{\mathtt{rv}}) \\
\text{(по леми 3.6)} \quad & = \sigma(var(\mathtt{qpat})) \cap \sigma(\overline{\mathtt{rv}}) \\
\text{(по дефиницији 3.18)} \quad & = \sigma(var(\mathtt{qpat}) \cap \overline{\mathtt{rv}}) \\
\text{(по леми 3.2)} \quad & = \sigma(\overline{\mathtt{rv}}) \\
\text{(како је } \sigma(\overline{\mathtt{rv}}) = \overline{rv}) \quad & = \overline{rv}
\end{aligned}$$

□

3.2.2 Моделовање проблема садржаности упита

Познато је да је проблем садржаности SPARQL упита неодлучив ако упит $\mathtt{Q}_2$ садржи пројекције [151]. Зато, уводи се претпоставка да је $\mathtt{Q}_2$ упит типа select*, односно да се све променљиве из његовог упит обрасца појављују у select клаузули. Проблем садржаности таквих упита своди се на следеће три провере:

(1) провера ваљаности формуле Θ (уведене дефиницијом 3.23),

(2) провера да ли важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$ (релација $\sim$ се уводи дефиницијом 3.24),

(3) провера ваљаности формуле Ψ (уведене дефиницијом 3.25).

Прецизније, закључује се да важи $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$ ако је задовољен један од следећа два услова:

- Провера (1) је задовољена. Интуитивно, овај услов одговара незадовољивости упита $\mathtt{Q}_1$, тј. по дефиницији 3.13, за било који скуп података $\mathbb{D}$, важи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} = \varnothing$. Зато, за било који упит $\mathtt{Q}_2$, важи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$.
- Обе провере (2) и (3) су задовољене. Интуитивно, ови услови проверавају да ли било које пресликавање из скупа $[\![\mathtt{Q}_1]\!]^{\mathbb{D}}$ припада такође и скупу $[\![\mathtt{Q}_2]\!]^{\mathbb{D}}$. Провера (2)

се односи на домен таквог пресликавања, тј. релевантне променљиве, док провера (3) испитује остатак.

Провера ваљаности формули Θ и Ψ врши се провером задовољивости њихових негација коришћењем SMT решавача.

Дефиниција 3.23 (Формула Θ) Нека је $\mathtt{Q}_1$ упит, нека $\overline{rv_1}$ означава променљиве из скупа $\mathcal{V}$ које одговарају релевантним променљивама упита $\mathtt{Q}_1$, и нека формула $\Phi_1(\overline{rv_1})$ одговара упиту $\mathtt{Q}_1$. Формула Θ дефинише се са:

$$\neg\big(\exists\overline{rv_1}\ \Phi_1(\overline{rv_1})\big)$$

Пример 3.11 На слици 3.6 дат је пример формуле Θ.

$$\neg\Big(\exists x_v \exists y_v \exists n_v \Big(\exists tel_v \big(\beta_d(x_v, a_i, UndergradStudent_i) \wedge \beta_d(x_v, takesCourse_i, y_v) \wedge \beta_d(x_v, name_i, n_v) \wedge \beta_d(x_v, phone_i, tel_v) \wedge n_v = Tim_l\big)\Big)\Big)$$

Слика 3.6: Формула Θ одговара упиту $\mathtt{Q}_1$ са слике 3.2 (лево). Ради читљивости, сваки IRI, литерал и променљива означени су одговарајућим индексом i, l и v, редом.

Дефиниција 3.24 (Релација $\sim$) Упити $\mathtt{Q}_1$ и $\mathtt{Q}_2$ су у релацији $\sim$, у ознаци $\mathtt{Q}_1 \sim \mathtt{Q}_2$, ако су скупови релевантних променљивих $\overline{\mathtt{rv}_1}$ и $\overline{\mathtt{rv}_2}$ ових упита једнаки, тј. ако важи

$$\overline{\mathtt{rv}_1} = \overline{\mathtt{rv}_2}.$$

Пример 3.12 Упити $\mathtt{Q}_1$ и $\mathtt{Q}_2$ са слике 3.2 јесу у релацији $\sim$, тј. важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$, јер су им скупови релевантних променљивих једнаки скупу $\{\mathtt{?x}, \mathtt{?n}, \mathtt{?y}\}$. Може се приметити да би релација важила и кад би у скуп пројектованих променљивих упита $\mathtt{Q}_2$ била додата променљива `?tel` (јер по дефиницији 3.12, она не би припадала скупу релевантних променљивих упита $\mathtt{Q}_2$), што не би важило и за упит $\mathtt{Q}_1$.

Лема 3.8 *Нека су $\mathtt{Q}_1$ и $\mathtt{Q}_2$ упити такви да $\mathtt{Q}_2$ не садржи пројекције и да важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$. Нека је $\overline{rv_1}$ скуп релевантних променљивих упита $\mathtt{Q}_1$, а $\mathtt{qpat}_2$ упит образац упита $\mathtt{Q}_2$. Тада важи:*

$$\overline{rv_1} = var(\mathtt{qpat}_2).$$

Доказ. По дефиницији 3.24, због $\mathtt{Q}_1 \sim \mathtt{Q}_2$, важи

$$\overline{\mathtt{rv}_1} = \overline{\mathtt{rv}_2}.$$

Тада, по леми 3.2, важи

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2}.$$

По дефиницији 3.11, како $\mathtt{Q}_2$ не садржи пројекције, важи

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2).$$

□

Дефиниција 3.25 (Формула Ψ) Нека су $\mathtt{Q}_1$ и $\mathtt{Q}_2$ упити и нека $\overline{rv_1}$ означава променљиве из скупа $\mathcal{V}$ које одговарају релевантним променљивама упита $\mathtt{Q}_1$. Нека формуле $\Phi_1(\overline{rv_1})$ и $\Phi_2(\overline{rv_1})$ одговарају упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$, редом. Формула Ψ дефинише се са:

$$\forall \overline{rv_1} \ \Big(\Phi_1(\overline{rv_1}) \Rightarrow \Phi_2(\overline{rv_1})\Big)$$

Пример 3.13 На слици 3.7 дат је пример формуле Ψ.

$$\begin{aligned}\forall x_v \forall y_v \forall n_v \Big(\Big(\exists tel_v \quad & (\beta_d(x_v, a_i, UndergradStudent_i) \wedge \beta_d(x_v, takesCourse_i, y_v) \wedge \\ & \beta_d(x_v, name_i, n_v) \wedge \quad \beta_d(x_v, phone_i, tel_v) \wedge \quad n_v = Tim_l)\Big) \\ \Rightarrow \Big(\quad & (\beta_d(x_v, a_i, UndergradStudent_i) \wedge \beta_d(x_v, takesCourse_i, y_v) \wedge \\ & \beta_d(x_v, name_i, n_v))\Big)\Big)\end{aligned}$$

Слика 3.7: Формула Ψ одговара упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$ са слике 3.2. Ради читљивости, сваки IRI, литерал и променљива означени су одговарајућим индексом i, l и v, редом.

Лема 3.9 *Нека су $\mathtt{Q}_1$ и $\mathtt{Q}_2$ упити и нека $\overline{rv_1}$ означава променљиве из скупа $\mathcal{V}$ које одговарају релевантним променљивама упита $\mathtt{Q}_1$. Нека формула $\Phi_1(\overline{rv_1})$ одговара упиту $\mathtt{Q}_1$. Нека су $\mathtt{qpat}_2$ и $\mathtt{D}_2$ упит образац и упитни скуп података упита $\mathtt{Q}_2$, редом. Ако $\mathtt{Q}_2$ не садржи пројекције и ако важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$, тада је формула Ψ једанка:*

$$\forall \overline{rv_1} \ \Big(\Phi_1(\overline{rv_1}) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\Big).$$

Доказ. По дефиницији 3.25, Ψ је дефинисано са

$$\forall \overline{rv_1} \ \Big(\Phi_1(\overline{rv_1}) \Rightarrow \Phi_2(\overline{rv_1})\Big),$$

тј. по дефиницији 3.22

$$\forall \overline{rv_1} \ \Big(\Phi_1(\overline{rv_1}) \Rightarrow \exists \overline{ov_2} \ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\Big),$$

где је $\overline{ov_2} = var(\sigma(\mathtt{qpat}_2)) \setminus \overline{rv_1}$. По дефиницији 3.24, како је $\mathtt{Q}_1 \sim \mathtt{Q}_2$, важи $\overline{\mathtt{rv}_1} = \overline{\mathtt{rv}_2}$, тј. по леми 3.2,

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2}.$$

Како упит $\mathtt{Q}_2$ не садржи пројекције, важи и

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2),$$

тј.

$$\mathtt{var}(\mathtt{qpat}_2) \setminus \overline{\mathtt{rv}_1} = \varnothing.$$

Зато, по дефиницији 3.18, важи

$$\sigma(\mathtt{var}(\mathtt{qpat}_2) \setminus \overline{\mathtt{rv}_1}) = \varnothing,$$

тј.

$$\sigma(\mathtt{var}(\mathtt{qpat}_2)) \setminus \sigma(\overline{\mathtt{rv}_1}) = \varnothing,$$

Даље, по леми 3.6, важи

$$var(\sigma(\mathtt{qpat}_2)) \setminus \overline{rv_1} = \varnothing.$$

Тада је $\overline{ov_2}$ једнако празном скупу, па се формула Ψ своди на

$$\forall \overline{rv_1} \left(\Phi_1(\overline{rv_1}) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2) \right).$$

□

У следећа два одељка доказана је сагласност (Теорема 3.18) и потпуност (Теорема 3.22) предложеног свођења проблема садржаности SPARQL упита.

3.2.3 Сагласност предложеног моделовања

Стандардна дефиниција $\mathcal{L}$-структуре над неком сигнатуром $\mathcal{L}$ дата је у литератури теорије модела, на пример [39, 133]. Следећа дефиниција представља њену конкретизацију за представљену SPARQL сигнатуру $\mathcal{L}$ која одговара упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$.

Дефиниција 3.26 ($\mathcal{L}$-структура $\mathfrak{D}$ која одговара упитима $\mathtt{Q}_1$, $\mathtt{Q}_2$ и скупу података $\mathbb{D}$) $\mathcal{L}$-структура која одговара скупу података $\mathbb{D}$ над SPARQL сигнатуром $\mathcal{L}$ која одговара упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$, у ознаци $\mathfrak{D}$, је уређен пар $(\mathcal{D}, \mathcal{I}^{\mathcal{D}})$, где:

- $\mathcal{D}$ је непразан домен једнак скупу

  $$\mathtt{IBLe} \cup (\sigma_{\mathtt{t}})^{-1}(\mathcal{C}),$$

  где I, B и L су скупови IRI-ја, неименованих чворова и литерала, редом, који се појављују у скупу података $\mathbb{D}$,

- $\mathcal{I}^{\mathcal{D}}$ је функција која пресликава нелогичке термове сигнатуре на следећи начин:

  – Интерпретација $\mathcal{I}^{\mathcal{D}}$ константи из $\mathcal{C}$ припада скупу IBL, и дефинисана је са $\mathcal{I}^{\mathcal{D}}(c) := (\sigma_{\mathtt{t}})^{-1}(c)$, за $c \in \mathcal{C}$.

  – Интерпретација $\mathcal{I}^{\mathcal{D}}$ предикатског симбила β_d је функција $B_d : \mathtt{IB} \times \mathtt{I} \times \mathtt{IBL} \to$ *bool*, дефинисана са:

  $$B_d(\mathtt{s}, \mathtt{p}, \mathtt{o}) = \top \text{ ако и само ако } (\mathtt{s}, \mathtt{p}, \mathtt{o}) \in df(\mathbb{D})$$

  – Интерпретација $\mathcal{I}^{\mathcal{D}}$ предикатског симбола β_n је функција $B_n : \mathtt{IB} \times \mathtt{I} \times \mathtt{IBL} \times \mathtt{I} \to$ *bool*, дефинисана са:

  $$B_n(\mathtt{s}, \mathtt{p}, \mathtt{o}, \mathtt{i}) = \top \text{ ако и само ако } (\mathtt{s}, \mathtt{p}, \mathtt{o}) \in gr_{\mathbb{D}}(\mathtt{i})$$

Како је функција $\sigma_{\mathtt{t}}$ бијективна по дефиницији 3.17, функција $(\sigma_{\mathtt{t}})^{-1}$ постоји и добро је дефинисана. Као последица ове дефиниције, ако је $B_n(\mathtt{s},\mathtt{p},\mathtt{o},\mathtt{i}) = \top$ онда $\mathtt{i} \in names(\mathbb{D})$. У супротном би $gr_{\mathbb{D}}(\mathtt{i})$ био једнак $\mathtt{G}_{\varnothing}$ и триплет $(\mathtt{s},\mathtt{p},\mathtt{o})$ не би припадао графу $gr_{\mathbb{D}}(\mathtt{i})$. Може се приметити да је домен $\mathcal{D}$ највише пребројив.

Валуација $\mathcal{L}$-формула у $\mathcal{L}$-структури $\mathfrak{D}$, обично означена са ν, је парцијална функција из скупа променљивих $\mathcal{V}$ у скуп IBL.

За $\mathcal{L}$-структуру $\mathfrak{M}$, валуацију ν, и формулу Υ из дате SPARQL сигнатуре $\mathcal{L}$, користи се стандардна нотација $(\mathfrak{M},\nu) \models \Upsilon$, која означава да је $\mathfrak{M}$ са валуацијом ν модел формуле Υ. Ако је формула Υ реченица, тј. ако не садржи слободне променљиве, такође се користи стандардна нотација $\mathfrak{M} \models \Upsilon$.

Дефиниција 3.27 (Модел $\mathcal{I}_\nu$) Нека је Υ формула над SPARQL сигнатуром $\mathcal{L}$, и нека је $(\mathfrak{D},\nu)$ њен модел, тј. $(\mathfrak{D},\nu) \models \Upsilon$. За функцију интерпретације од $(\mathfrak{D},\nu)$ дефинисану на стандардни начин у логици првог реда, користи се нотација $\mathcal{I}_\nu$, тј. $\mathcal{I}_\nu(\Upsilon)$ је тачно.

Ако формула Υ садржи променљиве $\overline{x}$, а валуација ν их пресликава у $\overline{c}$, тада $(\mathfrak{D},\nu) \models \Upsilon(\overline{c})$ означава да је $\mathcal{I}_\nu(\Upsilon(\overline{x}))$ тачно.

Дефиниција 3.28 (Релације $\rightarrow, \leftarrow, \leftrightarrow$) Нека је μ: VB $\rightarrow$ IBL пресликавање и ν: $\mathcal{V} \rightarrow$ IBL валуација.

- Пресликавање μ *дефинише* валуацију ν, у ознаци $\mu \rightarrow \nu$, ако је $dom(\nu)$ једнак $\sigma_{\mathtt{t}}(dom(\mu))$ и за свако $\mathtt{x} \in dom(\mu)$ важи $\nu(\sigma_{\mathtt{t}}(\mathtt{x})) = \mu(\mathtt{x})$.
- Пресликавање μ *је дефинисано* валуацијом ν, у ознаци $\mu \leftarrow \nu$, ако је $dom(\mu)$ једнак $\sigma_{\mathtt{t}}^{-1}(dom(\nu))$ и за свако $x \in dom(\nu)$ важи $\mu(\sigma_{\mathtt{t}}^{-1}(x)) = \nu(x)$.
- Пресликавање μ *одговара* валуацији ν, у ознаци $\mu \leftrightarrow \nu$, ако важи $\mu \rightarrow \nu$ и $\mu \leftarrow \nu$.

Релација $\leftrightarrow$ је добро дефинисана јер је функција $\sigma_{\mathtt{t}}$ бијективна. Може се приметити да ако важи $\mu \rightarrow \nu$, тада важи и $\mu \leftarrow \nu$, али и ако важи $\mu \leftarrow \nu$, важи и $\mu \rightarrow \nu$. Дакле, ове три релације су еквивалентне јер ако важи једна, важе све три.

Пример 3.14 Пресликавање μ, дефинисано са

$$\mu = \{\mathtt{?x} \mapsto \mathtt{:stud001}, \quad \mathtt{?n} \mapsto \mathtt{"Tim"}, \quad \mathtt{?y} \mapsto \mathtt{:course04}\},$$

одговара валуацији ν, дефинисаној са

$$\nu = \{x \mapsto \mathtt{:stud001}, \quad n \mapsto \mathtt{"Tim"}, \quad y \mapsto \mathtt{:course04}\},$$

тј. важи $\mu \leftrightarrow \nu$, јер је по дефиницији 3.17, $\sigma_{\mathtt{t}}$ повезује одговарајуће променљиве:

$$\mathtt{?x} \mapsto x, \quad \mathtt{?n} \mapsto n \quad \text{и} \quad \mathtt{?y} \mapsto y.$$

Слично нотацији $\mu_{\mathtt{x}\rightarrow\mathtt{c}}$, уводи се нотација $\nu_{x\rightarrow\mathtt{c}}$ која означава валуацију за коју важи $dom(\nu_{x\rightarrow\mathtt{c}}) := \{x\}$ и $\nu_{x\rightarrow\mathtt{c}}(x) := \mathtt{c}$. Може се приметити, по дефиницији 3.28, да важи $\mu_{\mathtt{x}\rightarrow\mathtt{c}} \leftrightarrow \nu_{x\rightarrow\mathtt{c}}$, јер $dom(\nu) = \{x\} = \{\sigma(\mathtt{x})\} = \sigma(\{\mathtt{x}\}) = \sigma(dom(\mu))$, и $\nu(x) = \mathtt{c} = \mu(\mathtt{x})$.

Дефиниција 3.3 може се такође користити и у контексту валуација: ν_1 и ν_2 су компатибилне, у ознаци $\nu_1 \simeq \nu_2$, ако за свако x такво да $x \in dom(\nu_1) \cap dom(\nu_2)$ важи $\nu_1(x) = \nu_2(x)$.

Лема 3.10 *Нека су* $\mu_1, \mu_2 : \texttt{VB} \to \texttt{IBL}$ *пресликавања, и* $\nu_1, \nu_2 \colon \mathcal{V} \to \texttt{IBL}$ *валуације такве да је* $\mu_1 \leftrightarrow \nu_1$ *и* $\mu_2 \leftrightarrow \nu_2$*. Тада важи:*

$$\mu_1 \simeq \mu_2$$
$$\textit{ако и само ако}$$
$$\nu_1 \simeq \nu_2.$$

Доказ. ($\Longrightarrow$) Ако је $dom(\nu_1) \cap dom(\nu_2)$ празан скуп, по дефиницији 3.3, ν_1 и ν_2 су компатибилне валуације, тј. $\nu_1 \simeq \nu_2$. У супротном, нека $x \in dom(\nu_1) \cap dom(\nu_2)$, тј.

$$x \in dom(\nu_1) \text{ и } x \in dom(\nu_2).$$

Из $\mu_1 \leftrightarrow \nu_1$ и $\mu_2 \leftrightarrow \nu_2$, по дефиницији 3.28, важи

$$x \in \{\sigma_{\texttt{t}}(\texttt{x}) \mid \texttt{x} \in dom(\mu_1)\} \text{ и } x \in \{\sigma_{\texttt{t}}(\texttt{x}) \mid \texttt{x} \in dom(\mu_2)\}.$$

Како је $\sigma_{\texttt{t}}$ бијективна функција, постоји јединствено $\texttt{x}$, такво да

$$x = \sigma_{\texttt{t}}(\texttt{x}),\ \texttt{x} \in dom(\mu_1) \text{ и } \texttt{x} \in dom(\mu_2)$$

тј.

$$\texttt{x} \in dom(\mu_1) \cap dom(\mu_2).$$

По дефиницији 3.3, из $\mu_1 \simeq \mu_2$, следи да важи

$$\mu_1(\texttt{x}) = \mu_2(\texttt{x}).$$

Из претпоставки $\mu_1 \leftrightarrow \nu_1$ и $\mu_2 \leftrightarrow \nu_2$, следи да важи

$$\nu_1(\sigma_{\texttt{t}}(\texttt{x})) = \mu_1(\texttt{x}) \text{ и } \mu_2(\texttt{x}) = \nu_2(\sigma_{\texttt{t}}(\texttt{x})).$$

Зато је

$$\nu_1(\sigma_{\texttt{t}}(\texttt{x})) = \nu_2(\sigma_{\texttt{t}}(\texttt{x})),\ \text{тј. } \nu_1(x) = \nu_2(x).$$

Из ове једнакости, по дефиницији 3.3 следи $\nu_1 \simeq \nu_2$.

($\Longleftarrow$) Ако је $dom(\mu_1) \cap dom(\mu_2)$ празан скуп, по дефиницији 3.3, μ_1 и μ_2 су компатибилна пресликавања, тј. $\mu_1 \simeq \mu_2$. У супротном, нека $\texttt{x} \in dom(\mu_1) \cap dom(\mu_2)$, тј.

$$\texttt{x} \in dom(\mu_1) \text{ и } x \in dom(\mu_2).$$

Из $\mu_1 \leftrightarrow \nu_1$ и $\mu_2 \leftrightarrow \nu_2$, по дефиницији 3.28, важи

$$\texttt{x} \in \{(\sigma_{\texttt{t}})^{-1}(x) \mid x \in dom(\nu_1)\} \text{ и } \texttt{x} \in \{(\sigma_{\texttt{t}})^{-1}(x) \mid x \in dom(\nu_2)\}.$$

Како је $(\sigma_{\texttt{t}})^{-1}$ бијективна функција, постоји јединствено x, такво да

$$\texttt{x} = (\sigma_{\texttt{t}})^{-1}(x),\ x \in dom(\nu_1) \text{ и } x \in dom(\nu_2)$$

тј.

$$x \in dom(\nu_1) \cap dom(\nu_2).$$

По дефиницији 3.3, из $\nu_1 \simeq \nu_2$, следи да важи

$$\nu_1(x) = \nu_2(x).$$

Из претпоставки $\mu_1 \leftrightarrow \nu_1$ и $\mu_2 \leftrightarrow \nu_2$, следи да важи

$$\mu_1((\sigma_{\mathtt{t}})^{-1}(x)) = \nu_1(x) \text{ и } \nu_2(x) = \mu_2((\sigma_{\mathtt{t}})^{-1}(x)).$$

Зато је

$$\mu_1((\sigma_{\mathtt{t}})^{-1}(x)) = \mu_2((\sigma_{\mathtt{t}})^{-1}(x)), \text{ тј. } \mu_1(\mathtt{x}) = \mu_2(\mathtt{x}).$$

Из ове једнакости, по дефиницији 3.3, следи $\mu_1 \simeq \mu_2$. □

Лема 3.11 *Нека су* $\mu, \mu_1, \mu_2 : \mathtt{VB} \to \mathtt{IBL}$ *пресликавања, и* $\nu, \nu_1, \nu_2 : \mathcal{V} \to \mathtt{IBL}$ *валуације такве да је* $\mu \leftrightarrow \nu$, $\mu_1 \leftrightarrow \nu_1$ *и* $\mu_2 \leftrightarrow \nu_2$. *Тада важи:*

$$\mu_1 \simeq \mu_2 \text{ и } \mu = \mu_1 \cup \mu_2$$
ако и само ако
$$\nu_1 \simeq \nu_2 \text{ и } \nu = \nu_1 \cup \nu_2.$$

Доказ. ($\Longrightarrow$) Из $\mu_1 \leftrightarrow \nu_1$, $\mu_2 \leftrightarrow \nu_2$ и $\mu_1 \simeq \mu_2$, по леми 3.10 следи

$$\nu_1 \simeq \nu_2,$$

па је $\nu_1 \cup \nu_2$ добро дефинисана валуација. Из $\mu \leftrightarrow \nu$, $\mu_1 \leftrightarrow \nu_1$ и $\mu \simeq \mu_1$ (јер је пресликавање μ проширење пресликавања μ_1), по леми 3.10 важи

$$\nu \simeq \nu_1.$$

Слично, из $\mu \leftrightarrow \nu$, $\mu_2 \leftrightarrow \nu_1$, и $\mu \simeq \mu_2$ (јер је μ проширење пресликавања μ_2), по леми 3.10, важи

$$\nu \simeq \nu_2.$$

Такође, важи и

$$\begin{aligned}
dom(\nu) &= \\
\text{(по дефиницији 3.28, јер је } \mu \leftrightarrow \nu) &= \sigma_{\mathtt{t}}(dom(\mu)) \\
\text{(како је } \mu = \mu_1 \cup \mu_2) &= \sigma_{\mathtt{t}}(dom(\mu_1) \cup dom(\mu_2)) \\
\text{(по дефиницији 3.18)} &= \sigma_{\mathtt{t}}(dom(\mu_1)) \cup \sigma_{\mathtt{t}}(dom(\mu_2)) \\
\text{(по дефиницији 3.28, јер је } \mu_1 \leftrightarrow \nu_1 \text{ и } \mu_2 \leftrightarrow \nu_2) &= dom(\nu_1) \cup dom(\nu_2) \\
\text{(јер је } \nu_1 \simeq \nu_2) &= dom(\nu_1 \cup \nu_2)
\end{aligned}$$

Како је $dom(\nu) = dom(\nu_1) \cup dom(\nu_2)$, за свако $x \in dom(\nu)$ важи

$$x \in dom(\nu_1) \text{ или } x \in dom(\nu_2).$$

Из компатибилности валуације ν са ν_1 и ν_2 редом, по дефиницији 3.3, важи

$$\nu(x) = \nu_1(x) \text{ или } \nu(x) = \nu_2(x).$$

Зато, како је $\nu_1 \simeq \nu_2$,

$$x \in dom(\nu) \text{ повлачи } \nu(x) = (\nu_1 \cup \nu_2)(x).$$

Коначно, из последње једнакости и из $dom(\nu) = dom(\nu_1 \cup \nu_2)$, следи да важи

$$\nu = \nu_1 \cup \nu_2.$$

($\Longleftarrow$) Из $\mu_1 \leftrightarrow \nu_1$, $\mu_2 \leftrightarrow \nu_2$ и $\nu_1 \simeq \nu_2$, по леми 3.10 следи

$$\mu_1 \simeq \mu_2,$$

па је $\mu_1 \cup \mu_2$ добро дефинисано пресликавање. Из $\mu \leftrightarrow \nu$, $\mu_1 \leftrightarrow \nu_1$ и $\nu \simeq \nu_1$ (јер је валуација ν проширење валуације ν_1), по леми 3.10 важи

$$\mu \simeq \mu_1.$$

Слично, из $\mu \leftrightarrow \nu$, $\mu_2 \leftrightarrow \nu_1$, и $\nu \simeq \nu_2$ (јер је валуација ν проширење валуације ν_2), по леми 3.10, важи

$$\mu \simeq \mu_2.$$

Такође, важи и

$$\begin{aligned}
& dom(\mu) && = \\
&\text{(по дефиницији 3.28, јер је } \mu \leftrightarrow \nu) && = (\sigma_{\mathtt{t}})^{-1}(dom(\nu)) \\
&\text{(како је } \nu = \nu_1 \cup \nu_2) && = (\sigma_{\mathtt{t}})^{-1}(dom(\nu_1) \cup dom(\nu_2)) \\
&\text{(по дефиницији 3.18)} && = (\sigma_{\mathtt{t}})^{-1}(dom(\nu_1)) \cup \\
& && \quad (\sigma_{\mathtt{t}})^{-1}(dom(\nu_2)) \\
&\text{(по дефиницији 3.28, јер је } \mu_1 \leftrightarrow \nu_1 \text{ и } \mu_2 \leftrightarrow \nu_2) && = dom(\mu_1) \cup dom(\mu_2) \\
&\text{(јер је } \mu_1 \simeq \mu_2) && = dom(\mu_1 \cup \mu_2)
\end{aligned}$$

Како је $dom(\mu) = dom(\mu_1) \cup dom(\mu_2)$, за свако $\mathtt{x} \in dom(\mu)$ важи

$$\mathtt{x} \in dom(\mu_1) \text{ или } \mathtt{x} \in dom(\mu_2).$$

Из компатибилности пресликавања μ са μ_1 и μ_2 редом, по дефиницији 3.3, важи

$$\mu(\mathtt{x}) = \mu_1(\mathtt{x}) \text{ или } \mu(\mathtt{x}) = \mu_2(x).$$

Зато, како је $\mu_1 \simeq \mu_2$,

$$\mathtt{x} \in dom(\mu) \text{ повлачи } \mu(\mathtt{x}) = (\mu_1 \cup \mu_2)(\mathtt{x}).$$

Коначно, из последње једнакости и из $dom(\mu) = dom(\mu_1 \cup \mu_2)$, следи да важи

$$\mu = \mu_1 \cup \mu_2.$$

□

Дефиниција 3.29 (Нотација $[[\cdot]]_\nu$) Нека је ν валуација, и t терм из SPARQL сигнатуре $\mathcal{L}$. Вредност терма t у валуацији ν, у ознаци $[[t]]_\nu$ је вредност из скупа `IBLe`, дефинисана са:

$$[[t]]_\nu := \begin{cases} \mathtt{err}\,, & t \in \mathcal{V} \text{ и } t \notin dom(\nu) \\ \mathcal{I}_\nu(t)\,, & \text{иначе} \end{cases}$$

Лема 3.12 *Нека је `E` израз, $\mu : VB \to IBL$ пресликавање и ν: $\mathcal{V} \to IBL$ валуација таква да је $\mu \leftrightarrow \nu$. Тада важи:*

$$[[E]]_\mu = [[\sigma(E)]]_\nu.$$

Доказ. Лема је доказана индукцијом по изразу E.

E је c, $\mathtt{c} \in \mathtt{IL}$:

	$[[\mathtt{c}]]_\mu$	$=$
(по дефиницији 3.8)		$= \mathtt{c}$
(по дефиницији 3.17)		$= (\sigma_\mathtt{t})^{-1}(\sigma_\mathtt{t}(\mathtt{c}))$
(по дефиницији 3.26)		$= \mathcal{I}^\mathcal{D}(\sigma_\mathtt{t}(\mathtt{c}))$
(по дефиницији 3.20)		$= \mathcal{I}^\mathcal{D}(\sigma(\mathtt{c}))$
(по дефиницији 3.29)		$= [[\sigma(\mathtt{c})]]_\nu$

E је x, $\mathtt{x} \in \mathtt{V}$:

- $\mathtt{x} \in dom(\mu)$
  Из $\mu \leftrightarrow \nu$ и $\mathtt{x} \in dom(\mu)$, по дефиницији 3.28, следи $\sigma(\mathtt{x}) \in dom(\nu)$.

	$[[\mathtt{x}]]_\mu$	$=$
(по дефиницији 3.8, јер $\mathtt{x} \in dom(\mu)$)		$= \mu(\mathtt{x})$
(по дефиницији 3.28, јер је $\mu \leftrightarrow \nu$)		$= \nu(\sigma_\mathtt{t}(\mathtt{x}))$
(по дефиницији 3.20)		$= \nu(\sigma(\mathtt{x}))$
(по дефиницији 3.27)		$= \mathcal{I}_\nu(\sigma(\mathtt{x}))$
(по дефиницији 3.29, јер $\sigma(\mathtt{x}) \in dom(\nu)$)		$= [[\sigma(\mathtt{x})]]_\nu$

- $\mathtt{x} \notin dom(\mu)$
  Из $\mu \leftrightarrow \nu$ и $\mathtt{x} \notin dom(\mu)$, по дефиницији 3.28, следи $\sigma(\mathtt{x}) \notin dom(\nu)$.

	$[[\mathtt{x}]]_\mu$	$=$
(по дефиницији 3.8, јер $\mathtt{x} \notin dom(\mu)$)		$= \mathtt{err}$
(по дефиницији 3.29, јер $\sigma(\mathtt{x}) \notin dom(\nu)$)		$= [[\sigma(\mathtt{x})]]_\nu$

□

Лема 3.13 *Нека је* E *израз и* x *променљива. Нека је* $\mu : VB \to IBL$ *пресликавање и* $\nu\colon \mathcal{V} \to IBL$ *валуација таква да је* $\mu \leftrightarrow \nu$. *Тада важи:*

$$\mu(x) = [[E]]_\mu$$
ако и само ако
$$(\mathfrak{D}, \nu) \models \sigma(x) = \sigma(E).$$

Доказ.

$\mu(\mathtt{x}) = [[\mathtt{E}]]_\mu$	акко
(по леми 3.12)	акко $\mu(\mathtt{x}) = [[\sigma(\mathtt{E})]]_\nu$
(по дефиницији 3.29)	акко $\mu(\mathtt{x}) = \mathcal{I}_\nu(\sigma(\mathtt{E}))$
(по дефиницији 3.28, јер је $\mu \leftrightarrow \nu$)	акко $\nu(\sigma(\mathtt{x})) = \mathcal{I}_\nu(\sigma(\mathtt{E}))$
(по дефиницији 3.27)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{x}) = \sigma(\mathtt{E})$

□

Лема 3.14 *Нека је* $\mu : \mathtt{VB} \to \mathtt{IBL}$ *пресликавање и* $\nu\colon \mathcal{V} \to \mathtt{IBL}$ *валуација таква да је* $\mu \leftrightarrow \nu$. *Тада важи:*

$$\mu \Vdash \mathtt{R}$$
ако и само ако
$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}).$$

Доказ. Лема је доказана индукцијом по услову R.

R је $\mathtt{E}_1 = \mathtt{E}_2$:

	$\mu \Vdash \mathtt{E}_1 = \mathtt{E}_2$ акко
(по дефиницији 3.9)	акко $[[\mathtt{E}_1]]_\mu \neq \mathtt{err}$ и $[[\mathtt{E}_2]]_\mu \neq \mathtt{err}$ и $[[\mathtt{E}_1]]_\mu = [[\mathtt{E}_2]]_\mu$
(по леми 3.12)	акко $[[\sigma(\mathtt{E}_1)]]_\nu \neq \mathtt{err}$ и $[[\sigma(\mathtt{E}_2)]]_\nu \neq \mathtt{err}$ и $[[\sigma(\mathtt{E}_1)]]_\nu = [[\sigma(\mathtt{E}_2)]]_\nu$
(по дефиницији 3.27 и 3.29)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{E}_1) = \sigma(\mathtt{E}_2)$
(по дефиницији 3.20)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{E}_1 = \mathtt{E}_2)$

R је $!\mathtt{R}_1$:

	$\mu \Vdash\ !\mathtt{R}_1$ акко
(по дефиницији 3.9)	акко не важи $\mu \Vdash \mathtt{R}_1$
(по индуктивној хипотези)	акко не важи $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}_1)$
(по дефиницији 3.27)	акко $(\mathfrak{D}, \nu) \models \neg\sigma(\mathtt{R}_1)$
(по дефиницији 3.20)	акко $(\mathfrak{D}, \nu) \models \sigma(\,!\mathtt{R}_1)$

R је $\mathtt{R}_1 \&\& \mathtt{R}_2$:

	$\mu \Vdash \mathtt{R}_1 \&\& \mathtt{R}_2$ акко
(по дефиницији 3.9)	акко $\mu \Vdash \mathtt{R}_1$ и $\mu \Vdash \mathtt{R}_2$
(по индуктивној хипотези)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}_1)$ и $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}_2)$
(по дефиницији 3.27)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}_1) \wedge \sigma(\mathtt{R}_2)$
(по дефиницији 3.20)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R}_1 \&\& \mathtt{R}_2)$

R је $\mathtt{R}_1 \,||\, \mathtt{R}_2$:
Доказ је аналоган претходном случају.

R је $(\mathtt{R}_1)$:
Доказ је аналоган случају $!\mathtt{R}_1$.

□

Лема 3.15 *Нека је* D *упитни скуп података,* G *граф у скупу података* D, *и* gp *граф образац. Нека је* $\mu : \mathtt{VB} \to \mathtt{IBL}$ *пресликавање и* $\nu\colon \mathcal{V} \to \mathtt{IBL}$ *валуација таква да је* $\mu \leftrightarrow \nu$. *Тада важи:*

$$\mu \in [\![gp]\!]^{D}_{G}$$
ако и само ако
$$(\mathfrak{D}, \nu) \models \sigma^{G}(gp) \text{ и } dom(\nu) = var(\sigma^{G}(gp)).$$

Доказ. По леми 3.1, из $\mu \in [\![\texttt{gp}]\!]^{\texttt{D}}_{\texttt{G}}$, важи $dom(\mu) = \texttt{var(gp)}$. Зато, по дефиницијама 3.20 и 3.28, како је $\mu \leftrightarrow \nu$, важи и

$$dom(\nu) = \sigma(dom(\mu)) = \sigma(\texttt{var(gp)}).$$

Тада, по леми 3.6, важи

$$dom(\nu) = var(\sigma(\texttt{gp})).$$

Остатак еквиваленције из формулације леме доказан је индукцијом по граф обрасцу gp.

gp је tp, где је tp триплет s p o:
($\Longrightarrow$) По дефиницији 3.10, из $\mu \in [\![\texttt{s p o}]\!]^{\texttt{D}}_{\texttt{G}}$ следи $[[\texttt{s p o}]]_\mu \in \texttt{G}$, тј. по дефиницији 3.8

$$[[\texttt{s}]]_\mu \ [[\texttt{p}]]_\mu \ [[\texttt{o}]]_\mu \in \texttt{G}.$$

По дефиницији 3.2, $\texttt{G} = \texttt{G}_\texttt{d}$, где је $\texttt{G}_\texttt{d} = df(\mathbb{D})$, или неко RDF обједињавање непразног скупа именованих графова, тј. $merge_{\texttt{D}}(\texttt{G}_{k_1}, ..., \texttt{G}_{k_m}), m > 0$, или именовани граф $\texttt{G}_{l_j}$, $j \in \{1, n\}$ у упитном скупу података D. Дакле, једна могућност од следеће три важи:

$$[[\texttt{s}]]_\mu \, [[\texttt{p}]]_\mu \, [[\texttt{o}]]_\mu \in \texttt{G}_\texttt{d} \tag{3.1}$$

$$[[\texttt{s}]]_\mu \, [[\texttt{p}]]_\mu \, [[\texttt{o}]]_\mu \in \texttt{G}_{k_j}, \text{ за неко } j \in \{1, ..., m\} \tag{3.2}$$

$$[[\texttt{s}]]_\mu \, [[\texttt{p}]]_\mu \, [[\texttt{o}]]_\mu \in \texttt{G}_{l_j}, \text{ за неко } j \in \{1, ..., n\}. \tag{3.3}$$

Из могућности (3.1), по дефиницији 3.26, важи

$$B_d([[\texttt{s}]]_\mu, [[\texttt{p}]]_\mu, [[\texttt{o}]]_\mu) = \top.$$

Тада, по леми 3.12, из $\mu \leftrightarrow \nu$, важи

$$B_d([[\sigma(\texttt{s})]]_\nu, [[\sigma(\texttt{p})]]_\nu, [[\sigma(\texttt{o})]]_\nu) = \top,$$

тј. по дефиницијама 3.26, 3.27 и 3.29:

$$(\mathfrak{D}, \nu) \models \beta_d(\sigma(\texttt{s}), \sigma(\texttt{p}), \sigma(\texttt{o})).$$

Из могућности (3.2), по дефиницији 3.19, скуп $cx(\texttt{G})$ садржи IRI константе из скупа $\mathcal{C}$ који одговарају графовима који чине граф G. Тада, постоји $i \in cx(\texttt{G})$, такво да i одговара графу који чини G, тј. i је једнако неком i_{k_j} из $cx(\texttt{G})$, коме одговара именовани граф $\texttt{G}_{k_j}$, тј. $gr_{\texttt{D}}(\texttt{i}) = \texttt{G}_{k_j}$. По дефиницији 3.26, из $[[\texttt{s}]]_\mu \ [[\texttt{p}]]_\mu \ [[\texttt{o}]]_\mu \in \texttt{G}_{k_j}$, важи

$$B_n([[\texttt{s}]]_\mu, [[\texttt{p}]]_\mu, [[\texttt{o}]]_\mu, i) = \top.$$

Из могућности (3.3), по дефиницији 3.19, постоји константа $i = i_{l_j}$, која одговара именованом графу $\texttt{G}_{l_j}$, тј. $gr_{\texttt{D}}(\texttt{i}) = \texttt{G}_{l_j}$. Може се приметити да $i \in cx(\texttt{G})$. По дефиницији 3.26, из $[[\texttt{s}]]_\mu \, [[\texttt{p}]]_\mu \, [[\texttt{o}]]_\mu \in \texttt{G}_{l_j}$, важи (исто као у претходној могућности):

$$B_n([[\texttt{s}]]_\mu, [[\texttt{p}]]_\mu, [[\texttt{o}]]_\mu, i) = \top.$$

Зато, по леми 3.12, из $\mu \leftrightarrow \nu$, важи

$$B_n([[\sigma(\mathtt{s})]]_\nu, [[\sigma(\mathtt{p})]]_\nu, [[\sigma(\mathtt{o})]]_\nu, i) = \top.$$

По дефиницијама 3.26, 3.27 и 3.29, важи

$$(\mathfrak{D}, \nu) \models \beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i),$$

тј.

$$(\mathfrak{D}, \nu) \models \bigvee_{i_j \in cx(\mathtt{G})} \beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i_j).$$

Објединивши све опције заједно, по дефиницији 3.20, важи

$$(\mathfrak{D}, \nu) \models \sigma^{\mathtt{G}}(\mathtt{s}\ \ \mathtt{p}\ \ \mathtt{o}).$$

($\Longleftarrow$) По дефиницији 3.20, из $(\mathfrak{D}, \nu) \models \sigma^{\mathtt{G}}(\mathtt{s}\ \ \mathtt{p}\ \ \mathtt{o})$, једна опција од могуће две важи:

$$(\mathfrak{D}, \nu) \models \beta_d(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o})), \qquad \text{ако } \mathtt{G} \notin dom(cx) \tag{3.4}$$

$$(\mathfrak{D}, \nu) \models \bigvee_{i_j \in cx(\mathtt{G})} \beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i_j), \text{ ако } \mathtt{G} \in dom(cx) \tag{3.5}$$

Из могућности (3.4), по дефиницијама 3.26, 3.27 и 3.29, важи

$$B_d([[\sigma(\mathtt{s})]]_\nu, [[\sigma(\mathtt{p})]]_\nu, [[\sigma(\mathtt{o})]]_\nu) = \top.$$

Зато, по леми 3.12, из $\mu \leftrightarrow \nu$, важи

$$B_d([[\mathtt{s}]]_\mu, [[\mathtt{p}]]_\mu, [[\mathtt{o}]]_\mu) = \top.$$

Тада, по дефиницији 3.26, важи

$$[[\mathtt{s}]]_\mu\, [[\mathtt{p}]]_\mu\, [[\mathtt{o}]]_\mu \in df(\mathbb{D}),$$

тј. како $\mathtt{G} \notin dom(cx)$,

$$[[\mathtt{s}]]_\mu\, [[\mathtt{p}]]_\mu\, [[\mathtt{o}]]_\mu \in \mathtt{G}.$$

Из могућности (3.5), постоји $i_j \in cx(\mathtt{G})$, $gr_{\mathtt{D}}(\mathtt{i}_j) = \mathtt{G}_j$, такво да

$$(\mathfrak{D}, \nu) \models \beta_n(\sigma(\mathtt{s}), \sigma(\mathtt{p}), \sigma(\mathtt{o}), i_j).$$

По дефиницијам 3.26, 3.27 и 3.29, важи:

$$B_n([[\sigma(\mathtt{s})]]_\nu, [[\sigma(\mathtt{p})]]_\nu, [[\sigma(\mathtt{o})]]_\nu, i_j) = \top.$$

Тада, по леми 3.12, из $\mu \leftrightarrow \nu$, важи

$$B_n([[\mathtt{s}]]_\mu, [[\mathtt{p}]]_\mu, [[\mathtt{o}]]_\mu, i_j) = \top.$$

По дефиницији 3.26, важи

$$[[\mathtt{s}]]_\mu\ \ [[\mathtt{p}]]_\mu\ \ [[\mathtt{o}]]_\mu \in gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i_j)),$$

тј.

$$[[\mathtt{s}]]_\mu\ \ [[\mathtt{p}]]_\mu\ \ [[\mathtt{o}]]_\mu \in \mathtt{G}_j.$$

По дефиницији 3.8, важи $[[\mathtt{s}\ \ \mathtt{p}\ \ \mathtt{o}]]_\mu \in \mathtt{G}_j$ за неко $\mathtt{G}_j$ које чини $\mathtt{G}$, тј. $[[\mathtt{s}\ \ \mathtt{p}\ \ \mathtt{o}]]_\mu \in \mathtt{G}$. Објединивши све могућности, по дефиницији 3.10, важи

$$\mu \in [\![\mathtt{s}\ \ \mathtt{p}\ \ \mathtt{o}]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

$\mathtt{gp}$ је $\mathtt{gp}_1.\mathtt{gp}_2$:

($\Longrightarrow$) По дефиницији 3.10, из $\mu \in [\![\mathtt{gp}_1.\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}$ следи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

По дефиницији 3.4, важи $\mu = \mu_1 \cup \mu_2$, где су μ_1 и μ_2 нека пресликавања за која важи

$$\mu_1 \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}},\ \mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}} \text{ и } \mu_1 \simeq \mu_2.$$

Применом индуктивне хипотезе на граф обрасце $\mathtt{gp}_1$ и $\mathtt{gp}_2$, пресликавања μ_1 и μ_2, и њихове одговарајуће валуације ν_1 и ν_2, редом, где је $\mu_1 \leftrightarrow \nu_1$ и $\mu_2 \leftrightarrow \nu_2$, важи:[5]

$$(\mathfrak{D}, \nu_1) \models \sigma(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu_2) \models \sigma(\mathtt{gp}_2).$$

По леми 3.11, из $\mu_1 \simeq \mu_2$, $\mu = \mu_1 \cup \mu_2$, $\mu_1 \leftrightarrow \nu_1$, $\mu_2 \leftrightarrow \nu_2$ и $\mu \leftrightarrow \nu$, важи $\nu_1 \simeq \nu_2$ и $\nu = \nu_1 \cup \nu_2$. Зато, ν је проширење и валуације ν_1 и валуације ν_2, па важи

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_2),$$

тј. по дефиницији 3.27,

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{gp}_2).$$

Тада, по дефиницији 3.20, важи

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1.\mathtt{gp}_2).$$

($\Longleftarrow$) По дефиницији 3.20, из $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1.\mathtt{gp}_2)$, важи

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{gp}_2),$$

тј. по дефиницији 3.27,

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_2).$$

Све слободне променљиве које се појављују у ове две формуле припадају скуповима $var(\sigma(\mathtt{gp}_1))$ и $var(\sigma(\mathtt{gp}_2))$ редом, па се рестрикције валуације ν на те домене означавају ν_1 и ν_2, редом. Може се приметити да је

$$dom(\nu) = dom(\nu_1) \cup dom(\nu_2) \text{ и } \nu = \nu_1 \cup \nu_2.$$

Зато се модели $(\mathfrak{D}, \nu_1)$ и $(\mathfrak{D}, \nu_2)$ могу користити у претходним формулама:

$$(\mathfrak{D}, \nu_1) \models \sigma(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu_2) \models \sigma(\mathtt{gp}_2).$$

Применом индуктивне хипотезе на граф обрасце $\mathtt{gp}_1$ и $\mathtt{gp}_2$, валуације ν_1 и ν_2, и њихова одговарајућа пресликавања μ_1 и μ_2, редом, где је $\mu_1 \leftrightarrow \nu_1$ и $\mu_1 \leftrightarrow \nu_1$, важи

$$\mu_1 \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \text{ и } \mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

По леми 3.11, из $\nu_1 \simeq \nu_2$, $\nu = \nu_1 \cup \nu_2$, $\mu_1 \leftrightarrow \nu_1$, $\mu_2 \leftrightarrow \nu_2$ и $\mu \leftrightarrow \nu$, важи

$$\mu_1 \simeq \mu_2 \text{ и } \mu = \mu_1 \cup \mu_2.$$

Тада, по дефиницији 3.4, важи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}},$$

тј. по дефиницији 3.10,

$$\mu \in [\![\mathtt{gp}_1.\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

[5] Ради једноставности, до краја овог доказа користи се нотација σ уместо σ^{G}.

gp је $\mathtt{gp}_1$ `filter R`

$\mu \in [\![\mathtt{gp}_1 \ \mathtt{filter\ R}]\!]^{\mathtt{D}}_{\mathtt{G}}$	акко
(по дефиницији 3.10)	акко $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}$ и $\mu \Vdash \mathtt{R}$
(по индуктивној хипотези)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma(\mathtt{gp}_1))$ и $\mu \Vdash \mathtt{R}$
(по леми 3.14)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma(\mathtt{gp}_1))$ и $(\mathfrak{D}, \nu) \models \sigma(\mathtt{R})$
(по дефиницији 3.27)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{R})$ и $dom(\nu) = var(\sigma(\mathtt{gp}_1))$
(по дефиницији 3.7, јер је $\mathtt{var(R)} \subseteq \mathtt{var}(\mathtt{gp}_1)$)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \wedge \sigma(\mathtt{R})$ и $dom(\nu) = var(\sigma(\mathtt{gp}))$
(по дефиницији 3.20)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1 \ \mathtt{filter\ R})$ и $dom(\nu) = var(\sigma(\mathtt{gp}))$

gp је $\{\mathtt{gp}_1\}$

$\mu \in [\![\{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$	акко
(по дефиницији 3.10)	акко $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}$
(по индуктивној хипотези)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma(\mathtt{gp}_1))$
(по дефиницији 3.7)	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma(\{\mathtt{gp}_1\}))$
(по дефиницији 3.20)	акко $(\mathfrak{D}, \nu) \models \sigma(\{\mathtt{gp}_1\})$ и $dom(\nu) = var(\sigma(\{\mathtt{gp}_1\}))$

□

Лема 3.16 *Нека је* $\mathbb{D}$ RDF *скуп података. Нека је* `Q` *упит,* $\overline{rv}$ *скуп његових релевантних променљивих и* $\Phi(\overline{rv})$ *одговарајућа формула. Нека је* $\mu : \mathtt{VB} \to \mathtt{IBL}$ *пресликавање и* $\nu : \mathcal{V} \to \mathtt{IBL}$ *валуација таква да је* $\mu \leftrightarrow \nu$*. Тада важи:*

$$\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$$
ако и само ако
$$(\mathfrak{D}, \nu) \models \Phi(\overline{rv}) \text{ и } dom(\nu) = var(\Phi(\overline{rv})).$$

Доказ. Нека је `D` упитни скуп података упита `Q`, `qpat` његов упит образац и `dv` скуп његових пројектованих променљивих.

($\Longrightarrow$) По дефиницији 3.12, из $\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$, следи $dom(\mu) = \overline{\mathtt{rv}}$. По дефиницији 3.28, како је $\mu \leftrightarrow \nu$, важи $dom(\nu) = \overline{rv}$. По дефиницијама 3.6 и 3.11, из $\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$, следи да постоји проширење μ' пресликавања μ, такво да

$$\mu' \in [\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По леми 3.15, за валуацију ν' такву да је $\mu' \leftrightarrow \nu'$, важи

$$(\mathfrak{D}, \nu') \models \sigma^{df(\mathtt{D})}(\mathtt{qpat}) \text{ и } dom(\nu') = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})).$$

Зато важи и

$$(\mathfrak{D}, \nu') \models \exists \overline{ov}\ \sigma^{df(\mathtt{D})}(\mathtt{qpat}),$$

тј. по дефиницији 3.22,

$$(\mathfrak{D}, \nu') \models \Phi(\overline{rv}).$$

По дефиницијама 3.5 и 3.28, из $\mu \leftrightarrow \nu$, $\mu' \leftrightarrow \nu'$ и $\mu \preceq \mu'$, следи $\nu \preceq \nu'$. По леми 3.7, важи $dom(\nu) = var(\Phi(\overline{rv})$. Дакле, ν је рестрикција валуације ν' дефинисана на свим слободним променљивама из $\Phi(\overline{rv})$, па онда важи и

$$(\mathfrak{D}, \nu) \models \Phi(\overline{rv}).$$

($\Longleftarrow$) По дефиницији 3.22, из $(\mathfrak{D}, \nu) \models \Phi(\overline{rv})$, следи

$$(\mathfrak{D}, \nu) \models \exists \overline{ov}\ \sigma^{df(\mathtt{D})}(\mathtt{qpat}).$$

Тада постоји проширење ν' валуације ν дефинисано на $dom(\nu) \cup \overline{ov}$, такво да важи

$$(\mathfrak{D}, \nu') \models \sigma^{df(\mathtt{D})}(\mathtt{qpat}).$$

Зато, из $dom(\nu) = var(\exists \overline{ov}\ \sigma^{df(\mathtt{D})}(\mathtt{qpat}))$, следи

$$dom(\nu') = var(\sigma^{df(\mathtt{D})}(\mathtt{qpat})).$$

По леми 3.15, за пресликавање μ' такво да је $\mu' \leftrightarrow \nu'$, важи

$$\mu' \in [\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По дефиницијама 3.5 и 3.28, из $\mu \leftrightarrow \nu$, $\mu' \leftrightarrow \nu'$ и $\nu \preceq \nu'$, следи $\mu \preceq \mu'$. Дакле, μ је рестрикција пресликавања μ' дефинисана на домену:

$dom(\mu)$	$=$
(по дефиницији 3.28, јер је $\mu \leftrightarrow \nu$)	$= (\sigma_{\mathtt{t}})^{-1}(dom(\nu))$
(по леми 3.7)	$= (\sigma_{\mathtt{t}})^{-1}(\overline{rv})$
(јер је $\sigma(\overline{\mathtt{rv}}) = \overline{rv}$)	$= \overline{\mathtt{rv}}$
(по леми 3.2)	$= \mathtt{var(qpat)} \cap \overline{\mathtt{dv}}$
(по леми 3.1)	$= dom(\mu') \cap \overline{\mathtt{dv}}$

Зато, по дефиницији 3.5, $\mu = \mu'_{\overline{\mathtt{dv}}}$, и по дефиницијама 3.6 и 3.11, важи

$$\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}.$$

□

Може се приметити да је пресликавање μ увек разматрано као пресликавање из скупа VB у скуп IBL. Међутим, у леми 3.16, $dom(\mu)$ је заправо подскуп од скупа V и не садржи неименоване чворове (према дефиницији 3.11).

Лема 3.17 *Ако је формула Θ ваљана, тада је упит Q_1 незадовољив.*

Доказ. Претпоставимо да је упит $\mathtt{Q}_1$ задовољив. По дефиницији 3.13, постоји скуп података $\mathbb{D}$ и пресликавање μ такво да $\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}$. Тада, по леми 3.16, за валуацију ν такву да је $\mu \leftrightarrow \nu$, важи

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1}),$$

где је $\overline{\mathtt{rv}_1}$ скуп релевантних променљивих упита $\mathtt{Q}_1$ и формула $\Phi_1(\overline{rv_1})$ одговара упиту $\mathtt{Q}_1$. Зато, важи и

$$(\mathfrak{D}, \nu) \models \exists \overline{rv_1}\ \Phi_1(\overline{rv_1}).$$

По дефиницији 3.23, из ваљаности формуле Θ, важи

$$(\mathfrak{D}, \nu) \models \neg\big(\exists \overline{rv_1}\ \Phi_1(\overline{rv_1})\big).$$

Дакле, $(\mathfrak{D}, \nu)$ је модел и формуле и њене негације, што је контрадикција, тј. упит $\mathtt{Q}_1$ је незадовољив. □

Теорема 3.18 (Сагласност) *Ако је*

(1)	Θ *ваљана,*
или је	
(2)	$\mathtt{Q}_1 \sim \mathtt{Q}_2$ *и* Ψ *је ваљана,*
тада важи	
	$\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$.

Доказ. Случај (1): Формула Θ је ваљана.
По леми 3.17, упит $\mathtt{Q}_1$ је незадовољив. По дефиницији 3.13, $[\![\mathtt{Q}_1]\!]^{\mathbb{D}}$ је празан скуп за било који скуп података $\mathbb{D}$. Дакле, важи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$, тј. по дефиницији 3.14,

$$\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2.$$

Случај (2): Важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$ и формула Ψ је ваљана.
Нека је $\mathbb{D}$ било који скуп података, и μ пресликавање такво да

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

По леми 3.16, за валуацију ν такву да је $\mu \leftrightarrow \nu$, важи

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1}) \text{ и } dom(\nu) = var(\Phi_1(\overline{rv_1})).$$

По леми 3.7, важи $dom(\nu) = \overline{rv_1}$. По леми 3.8, како је $\mathtt{Q}_1 \sim \mathtt{Q}_2$ и $\mathtt{Q}_2$ не садржи пројекције, важи

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2),$$

тј.

$$\sigma(\overline{\mathtt{rv}_1}) = \sigma(\mathtt{var}(\mathtt{qpat}_2)).$$

Тада, по леми 3.6, важи

$$\overline{rv_1} = var(\sigma(\mathtt{qpat}_2)).$$

Дакле, $dom(\nu)$ је једнак $var(\sigma(\mathtt{qpat}_2))$, такође. По леми 3.9, како је $\mathtt{Q}_1 \sim \mathtt{Q}_2$, $\mathtt{Q}_2$ не садржи пројекције и Ψ је ваљана, важи

$$(\mathfrak{D}, \nu) \models \forall \overline{rv_1}\ (\Phi_1(\overline{rv_1}) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)).$$

Како је ν дефинисана на свим слободним променљивама из $\Phi_1(\overline{rv_1}) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)$, важи:

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1}) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Даље, из $(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1})$, важи

$$(\mathfrak{D}, \nu) \models \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Тада, по леми 3.15, како је $\mu \leftrightarrow \nu$ и $dom(\nu) = var(\sigma(\mathtt{qpat}_2))$, важи

$$\mu \in [\![\mathtt{qpat}_2]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)},$$

тј. по дефиницији 3.11, како $\mathtt{Q}_2$ не садржи пројекције,

$$\mu \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

Дакле, сваки елемент скупа $[\![\mathtt{Q}_1]\!]^{\mathbb{D}}$ такође припада и скупу $[\![\mathtt{Q}_2]\!]^{\mathbb{D}}$ за било који скуп података $\mathbb{D}$, тј. $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$. По дефиницији 3.15, важи

$$\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2.$$

□

3.2.4 Потпуност предложеног моделовања

Стандардне дефиниције утапања, изоморфизама и подструктура дате су у литератури теорије модела, на пример [39, 133]. Следећа дефиниција представља њихову конкретизацију за представљену сигнатуру $\mathcal{L}$ која одговара упитима $\mathtt{Q}_1$ и $\mathtt{Q}_2$.

Дефиниција 3.30 ($\mathcal{L}$-утапање, $\mathcal{L}$-изоморфизам, релација $\cong$, $\mathcal{L}$-подструктура) Нека су $\mathcal{M}$ и $\mathcal{N}$ непразни домени. Нека су $\mathfrak{M} = (\mathcal{M}, \mathcal{I}^{\mathcal{M}})$ и $\mathfrak{N} = (\mathcal{N}, \mathcal{I}^{\mathcal{N}})$ $\mathcal{L}$-структуре које одговарају упитима $\mathtt{Q}_1$, $\mathtt{Q}_2$ и скупу података $\mathbb{D}$.

$\mathcal{L}$-утапање *$\mathcal{L}$-утапање* $\theta : \mathfrak{M} \to \mathfrak{N}$ је функција $\theta : \mathcal{M} \to \mathcal{N}$ која има следећа својства:

- θ је инјективна функција,
- за релационе симболе β_d и β_n сигнатуре $\mathcal{L}$, и за свако $s, p, o, i \in \mathcal{M}$ важи

$$\begin{aligned} (s,p,o) \in \mathcal{I}^{\mathcal{M}}(\beta_d) \text{ ако и само ако } (\theta(s), \theta(p), \theta(o)) &\in \mathcal{I}^{\mathcal{N}}(\beta_d), \text{ и} \\ (s,p,o,i) \in \mathcal{I}^{\mathcal{M}}(\beta_n) \text{ ако и само ако } (\theta(s), \theta(p), \theta(o), \theta(i)) &\in \mathcal{I}^{\mathcal{N}}(\beta_n). \end{aligned}$$

- за сваки симбол константе c сигнатуре $\mathcal{L}$ важи

$$\theta(\mathcal{I}^{\mathcal{M}}(c)) = \mathcal{I}^{\mathcal{N}}(c).$$

$\mathcal{L}$-изоморфизам *$\mathcal{L}$-изоморфизам* из $\mathfrak{M}$ у $\mathfrak{N}$ је бијективно $\mathcal{L}$-утапање из $\mathfrak{M}$ у $\mathfrak{N}$.

Релација $\cong$ $\mathcal{L}$-структуре $\mathfrak{M}$ и $\mathfrak{N}$ су *изоморфне*, у ознаци $\mathfrak{M} \cong \mathfrak{N}$, ако постоји $\mathcal{L}$-изоморфизам $\theta : \mathfrak{M} \to \mathfrak{N}$.

$\mathcal{L}$-подструктура $\mathcal{L}$-структура $\mathfrak{M}$ је *$\mathcal{L}$-подструктура* од $\mathcal{L}$-структуре $\mathfrak{N}$ ако је $\mathcal{M} \subseteq \mathcal{N}$ и функција $\iota : \mathcal{M} \to \mathcal{N}$, таква да је $\iota(a) = a$ за свако $a \in \mathcal{M}$, је $\mathcal{L}$-утапање.

Дефиниција $\mathcal{L}$-утапања треба да чува конгруенцију свих релационих симбола из $\mathcal{P}_s$. Дакле, друго својство треба да важи не само за β_d и β_n, већ и за симбол једанкости, али оно може бити доказано из првог својства, тј. из инјективности функције θ:

- за свако $m_1, m_2 \in \mathcal{M}$, ако је $m_1 = m_2$ тада важи $\theta(m_1) = \theta(m_2)$, јер је θ функција;
- обратно важи јер је θ инјективна.

Са друге стране, ако се друго ствојство формулише и за симбол једнакости, прво својство није неопходно, јер је оно директна последица другог, баш за симбол једнакости.

Лема 3.19 *Нека је Υ реченица из неке сигнатуре $\mathcal{L}$, и нека су $\mathfrak{D}'$ и $\mathfrak{D}''$ $\mathcal{L}$-структуре такве да је $\mathfrak{D}' \cong \mathfrak{D}''$. Тада важи:*

$$\mathfrak{D}' \models \Upsilon \quad \textit{ако и само ако} \quad \mathfrak{D}'' \models \Upsilon$$

Доказ. Доказ ове леме може се наћи у [133]. □

Лема 3.20 *Нека је $\mathfrak{D}' = (\mathcal{D}', \mathcal{I}^{\mathcal{D}'})$ нека $\mathcal{L}$-структура над* SPARQL *сигнатуром $\mathcal{L}$ која одговара упитима Q_1 и Q_2. Нека је Υ реченица над истом сигнатуром таква да је*

$$\mathfrak{D}' \models \Upsilon.$$

Тада постоји скуп података $\mathbb{D}$ и одговарајућа $\mathcal{L}$-структура $\mathfrak{D} = (\mathcal{D}, \mathcal{I}^{\mathcal{D}})$ над истом сигнатуром, таква да важи

$$\mathfrak{D} \models \Upsilon.$$

Доказ. Према дефиницији 3.30, $\mathfrak{D}'$ има непразан домен $\mathcal{D}'$, који може бити бесконачан. По *Löwenheim-Skolem* теореми (наниже), постоји $\mathcal{L}$-структура $\mathfrak{D}'' = (\mathcal{D}'', \mathcal{I}^{\mathcal{D}''})$ над истом сигнатуром, таква да је њен домен $\mathcal{D}''$ пребројив и да важи

$$\mathfrak{D}'' \models \Upsilon.$$

У наставку доказа дата је конструкција RDF скупа података $\mathbb{D}$ и одговарајуће $\mathcal{L}$-структуре $\mathfrak{D} = (\mathcal{D}, \mathcal{I}^{\mathcal{D}})$, тако да важи $\mathfrak{D}'' \cong \mathfrak{D}$. Прво, дефинише се функција $\theta : \mathcal{D}'' \to$ `IBLe` на следећи начин:

1. за свако $c'' \in \mathcal{D}''$ такво да је $c'' = \mathcal{I}^{\mathcal{D}''}(c)$ за неку константу из $\mathcal{C}$, $\theta(c'')$ је једнако $(\sigma_{\mathtt{t}})^{-1}(c)$. Може се приметити да је рестрикција функције θ на скуп оваквих елемената инјективна, по дефиницији 3.17.
2. за остале елементе x из $\mathcal{D}''$, $\theta(x)$ је једнако произвољном елементу из `IBL`, али водећи рачуна да функција θ остане инјективна. Ово је могуће, јер су `IBL` и $\mathcal{D}''$ пребројиви скупови.

Тада, конструише се RDF скуп података $\mathbb{D}$ на следећи начин:

- за сваку константу i из $\mathcal{D}''$ дефинише се граф $\mathtt{G}_i$, и функција $gr_{\mathbb{D}}$:

$$\mathtt{G}_i := \{ \big(\theta(s), \theta(p), \theta(o)\big) \mid s, p, o \in \mathcal{D}'' \text{ и } \mathcal{I}^{\mathcal{D}''}(\beta_n)(s, p, o, i) \text{ је тачно}\}$$
$$gr_{\mathbb{D}}(\theta(i)) := \mathtt{G}_i$$

- дефинише се граф $\mathtt{G}_{\mathtt{d}}$:

$$\mathtt{G}_{\mathtt{d}} := \{ \big(\theta(s), \theta(p), \theta(o)\big) \mid s, p, o \in \mathcal{D}'' \text{ и } \mathcal{I}^{\mathcal{D}''}(\beta_d)(s, p, o) \text{ је тачно}\}$$

Тада, скуп података $\mathbb{D}$ садржи подразумевани граф $\mathtt{G_d}$ и именоване графове $\mathtt{G}_i$ са одговарајућим именима $\theta(i)$. Може се приметити да су ови графови непразни јер формула која садржи моделовање упита има у себи предикате β_n и β_d, и за њу постоји модел у $\mathfrak{D}''$.
Функција θ је $\mathcal{L}$-утапање, по дефиницији 3.30, јер:

1. θ је инјективна, по својој конструкцији;
2. за свако $s, p, o, i \in \mathcal{D}''$, важи

$$\begin{aligned}
&\mathcal{I}^{\mathcal{D}''}(\beta_d)(s,p,o) && \text{акко}\\
&\text{(по конструкцији)} && \text{акко } (\,\theta(s),\theta(p),\theta(o)\,) \in \mathtt{G_d}\\
&\text{(по дефиницији 3.26)} && \text{акко } B_d(\theta(s),\theta(p),\theta(o))\\
&\text{(по дефиницији 3.26)} && \text{акко } \mathcal{I}^{\mathcal{D}}(\beta_d)(\theta(s),\theta(p),\theta(o)),
\end{aligned}$$

и

$$\begin{aligned}
&\mathcal{I}^{\mathcal{D}''}(\beta_n)(s,p,o,i) && \text{акко}\\
&\text{(по конструкцији)} && \text{акко } (\,\theta(s),\theta(p),\theta(o)\,) \in gr_{\,\mathbb{D}}(\theta(i))\\
&\text{(по дефиницији 3.26)} && \text{акко } B_n(\theta(s),\theta(p),\theta(o),\theta(i))\\
&\text{(по дефиницији 3.26)} && \text{акко } \mathcal{I}^{\mathcal{D}}(\beta_n)(\theta(s),\theta(p),\theta(o),\theta(i));
\end{aligned}$$

3. за сваки симбол константе c сигнатуре $\mathcal{L}$, $\mathcal{I}^{\mathcal{D}''}(c) \in \mathcal{D}''$, и $\theta(\mathcal{I}^{\mathcal{D}''}(c)) = \mathcal{I}^{\mathcal{D}}(c)$ по конструкцији.

Функција θ је сурјективна, јер за сваки елемент $\mathrm{x} \in \mathcal{D}$, постоји елемент $x \in \mathcal{D}''$ такав да је $\theta(x) = \mathrm{x}$:

- ако је x интерпретација константе из скупа $\mathcal{C}$, према дефиницији 3.26, конструкцији функције θ и чињенице да су $\mathfrak{D}''$ и $\mathfrak{D}$ одговарајуће $\mathcal{L}$-структуре, постоји елемент x такав да је $\theta(x) = \mathrm{x}$;
- у супротном, било који други елемент из $\mathcal{D}$ је слика неког елемента из $\mathcal{D}''$ по конструкцији функције θ и скупа података $\mathbb{D}$.

Зато, $\mathcal{L}$-утапање θ је бијективно, и по дефиницији 3.30, важи

$$\mathfrak{D}'' \cong \mathfrak{D}.$$

Тада, по леми 3.19, важи и

$$\mathfrak{D} \models \Upsilon.$$

□

Други смер леме 3.17 формулисан је у оквиру следеће леме.

Лема 3.21 *Ако је упит Q_1 незадовољив, тада је формула Θ ваљана.*

Доказ. Претпоставимо да формула Θ није ваљана. Тада постоји $\mathcal{L}$-структура $\mathfrak{D}' = (\mathcal{D}', \mathcal{I}^{\mathcal{D}'})$, таква да важи $\mathfrak{D}' \models \neg\Theta$. По дефиницији 3.23, $\neg\Theta$ је једнако

$$\exists \overline{rv_1}\ \Phi_1(\overline{rv_1}).$$

Може се приметити да је ова формула реченица (све променљиве су квантификоване). По леми 3.20, постоји скуп података $\mathbb{D}$ и одговарајућа $\mathcal{L}$-структура $\mathfrak{D} = (\mathcal{D}, \mathcal{I}^{\mathcal{D}})$, таква да важи

$$\mathfrak{D} \models \exists \overline{rv_1}\ \Phi_1(\overline{rv_1}).$$

Тада, постоји валуација ν дефинисана на $\overline{rv_1}$, таква да важи

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1}),$$

тј. по дефиницији 3.27,

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1}).$$

По леми 3.7, важи $dom(\nu) = var(\Phi_1(\overline{rv_1}))$. Даље, по леми 3.16, за пресликавање μ такво да је $\mu \leftrightarrow \nu$, важи

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

Ово је контрадикција са незадовољивошћу упита $\mathtt{Q}_1$. Дакле, формула Θ је ваљана. □

Теорема 3.22 (Потпуност) *Ако важи $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$, тада*

(1) Θ *је ваљана,*

или

(2) $\mathtt{Q}_1 \sim \mathtt{Q}_2$ *и* Ψ *је ваљана.*

Доказ. Случај (1): Упит $\mathtt{Q}_1$ је незадовољив.
По леми 3.21, из незадовољивости упита $\mathtt{Q}_1$ следи да је формула Θ ваљана.

Случај (2): Упит $\mathtt{Q}_1$ је задовољив.
У овом случају, биће доказана релација $\mathtt{Q}_1 \sim \mathtt{Q}_2$ и ваљаност формуле Ψ. Како је упит $\mathtt{Q}_1$ задовољив, постоји скуп података $\mathbb{D}$ и пресликавање μ такво да

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

По дефиницији 3.15, из $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$, следи $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$, тј.

$$\mu \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

По дефиницији 3.12, важи $dom(\mu) = \overline{\mathtt{rv}_1}$ и $dom(\mu) = \overline{\mathtt{rv}_2}$, па и $\overline{\mathtt{rv}_1} = \overline{\mathtt{rv}_2}$, тј. по дефиницији 3.24,

$$\mathtt{Q}_1 \sim \mathtt{Q}_2.$$

Претпоставимо да формула Ψ није ваљана. Тада постоји $\mathcal{L}$-структура $\mathfrak{D}' = (\mathcal{D}', \mathcal{I}^{\mathcal{D}'})$, таква да $\mathfrak{D}' \models \neg\Psi$. По леми 3.9, како важи $\mathtt{Q}_1 \sim \mathtt{Q}_2$ и $\mathtt{Q}_2$ не садржи пројекције, $\neg\Psi$ је једнако

$$\neg\Big(\forall\overline{rv_1}\big(\Phi_1(\overline{rv}_1) \Rightarrow \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\big)\Big), \tag{3.6}$$

где је $\mathtt{D}_2$ упитни скуп података упита $\mathtt{Q}_2$, тј.

$$\exists\overline{rv_1}\ \big(\Phi_1(\overline{rv}_1) \wedge \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\big). \tag{3.7}$$

Може се приметити да је ова формула реченица (све променљиве су квантификоване). По леми 3.20, постоји скуп података $\mathbb{D}$ и одговарајућа $\mathcal{L}$-структура $\mathfrak{D} = (\mathcal{D}, \mathcal{I}^{\mathcal{D}})$, таква да важи

$$\mathfrak{D} \models \exists\overline{rv_1}\ \big(\Phi_1(\overline{rv}_1) \wedge \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\big). \tag{3.8}$$

Тада, постоји валуација ν дефинисана на $\overline{rv_1}$, таква да важи

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv}_1) \wedge \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2), \tag{3.9}$$

тј. по дефиницији 3.27,

$$(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv}_1) \quad \text{и} \quad (\mathfrak{D}, \nu) \models \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2). \tag{3.10}$$

По леми 3.7, важи $dom(\nu) = var(\Phi_1(\overline{rv_1}))$. Тада, по леми 3.16, за пресликавање μ такво да је $\mu \leftrightarrow \nu$, важи

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

По леми 3.8, како је $\mathtt{Q}_1 \sim \mathtt{Q}_2$ и $\mathtt{Q}_2$ не садржи пројекције, важи

$$\overline{\mathtt{rv}_1} = \mathtt{var}(\mathtt{qpat}_2),$$

тј.

$$\sigma(\overline{\mathtt{rv}_1}) = \sigma(\mathtt{var}(\mathtt{qpat}_2)),$$

По леми 3.6, важи и

$$\overline{rv_1} = var(\sigma(\mathtt{qpat}_2)).$$

Дакле, $dom(\nu)$ је једнак и $var(\sigma(\mathtt{qpat}_2))$, па по леми 3.15, важи

$$\mu \notin [\![\mathtt{qpat}_2]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)}.$$

Тада, по дефиницији 3.11, како упит $\mathtt{Q}_2$ не садржи пројекције, важи

$$\mu \notin [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

Дакле, може се закључити да $[\![\mathtt{Q}_1]\!]^{\mathbb{D}} \subseteq [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$ не важи, тј. по дефиницији 3.15, не важи $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$. Ово је контрадикција са претпоставком теореме. Дакле, формула Ψ је ваљана. □

3.3 Подупити као граф обрасци

Подупити дозвољавају упите у оквиру других упита да би олакшали писање нових упита и поновно коришћење старих. Подупити су простији од упита јер не могу да садрже `from` и `from named` клаузуле, и користе се на месту граф образаца [8]. Да би обухватила овај тип конструкта, граматика SPARQL упита проширује се на начин приказан на слици 3.8, док је његова семантика дата следећом дефиницијом, која проширује дефиницију 3.10, па носи исту ознаку на коју је додат суфикс (п), тј. прво слово од речи подупит[6].

```
GPattern  ::= ...                    | SubQuery  ::= select Vars
            | '{' SubQuery '}'       |               where QPattern
```

Слика 3.8: Додавање подупита граматици са слике 3.1

Дефиниција 3.10(п) (Евалуација граф обрасца) ... Нека је $\mathtt{Q}_{\mathtt{sq}}$ подупит. Евалуација граф обрасца над графом `G` у скупу података `D`, у ознаци $[\![\cdot]\!]^{\mathtt{D}}_{\mathtt{G}}$, дефинисана је рекурзивно са:

$$\ldots := \ldots$$
$$[\![\{\mathtt{Q}_{\mathtt{sq}}\}]\!]^{\mathtt{D}}_{\mathtt{G}} := [\![\mathtt{Q}_{\mathtt{sq}}]\!]^{\mathtt{D}}$$

Пример 3.15 На слици 3.9 са леве стране, дат је упит $\mathtt{Q}_1$ који садржи подупит као граф образац, а са десне је њему еквивалентан упит $\mathtt{Q}_2$, који не садржи подупите. Упити враћају студенте основних студија, њихова имена, бројеве телефона и изабране курсеве, под условом да су сви подаци присутни у активном графу, да се студент зове `Tim` и да постоји податак о адреси његове електронске поште.

Евалуација SPARQL упита извршава се одоздо навише, што значи да се најугнежденији упит извршава први. Тада се резултати подупита пројектују назад спољашњем упиту, и само су му пројектоване промењљиве видљиве, тј. оне које нису пројектоване ван су опсега његових промењљивих. Следеће проширење дефиниције 3.7 прецизира овај услов.

Дефиниција 3.7(п) (Функција `var`) ... Промењљиве које се појављују у граф обрасцу `gp`, у ознаци `var(gp)`, су

$$\mathtt{var(gp)} := \begin{cases} \ldots, & \ldots \\ \underline{\mathtt{dv}_{\mathtt{sq}}}, & \mathtt{gp}\ \text{је}\ \{\mathtt{Q}_{\mathtt{sq}}\} \end{cases}$$

Пример 3.16 Промењљива `?z` из подупита у упиту $\mathtt{Q}_1$ са слике 3.9, није доступна ван подупита. Промењљива `?z` из упита $\mathtt{Q}_1$ која се налази ван подупита, је потпуно нова и

[6] Слична нотација постоји и у нумерисању осталих дефиниција из овог поглавља (којима се допуњује нека дефиниција из поглавља 3.1 или 3.2), као и дефиниција које следе у поглављима 3.4 и 3.5, где се користе суфикси (у) и (о), као прво слово оператора `union` и `optional`, редом.

Упит Q_1

```
select ?x ?n ?z ?y
where {
  {
    select ?x ?n
    where {
      ?x a      :UndergradStudent .
      ?x :name  ?n .
      ?x :email ?z
      filter (?n = "Tim")
    }
  } .
  ?x :takesCourse ?y .
  ?x :phone       ?z
}
```

Упит Q_2

```
select ?x ?n ?z ?y
where {
  ?x a             :UndergradStudent .
  ?x :name         ?n .
  ?x :email        ?z1 .
  ?x :takesCourse  ?y .
  ?x :phone        ?z
  filter (?n = "Tim")
}
```

Слика 3.9: Пример SPARQL упита са подупитом као граф обрасцем (лево) и њему еквивалентан упит без подупита (десно)

независна променљива, тј. не захтева се једнакост објеката из триплета `?x :email ?z` и `?x :phone ?z`. У упиту Q_2 са исте слике користе се различито именоване променљиве `?z` и `?z1`.

Дефиниција 3.31 представља специјалну форму граф образаца која олакшава свођење упита са подупитима на еквивалентне упите без подупита. Неформално, она захтева да су све `filter` клаузуле померене на крај граф образаца и да, ако постоји више њих, оне буду груписане у једну.

Дефиниција 3.31 (`filter`-нормална форма) Граф образац `gp` је у *`filter`-нормалној форми* ако:

(i) `gp` је триплет образац,

(ii) `gp` је $\{Q_{sq}\}$ и упит образац $qpat_{sq}$ подупита Q_{sq} је у `filter`-нормалној форми,

(iii) `gp` је $gp_1 . gp_2$, где су gp_1 и gp_2 у `filter`-нормалној форми (i), (ii) или (iii),

(iv) `gp` је gp_1 `filter R` и gp_1 је у `filter`-нормалној форми (i), (ii) или (iii).

Следећа лема показује да се сваки граф образац (разматран до сада) може свести на еквивалентан граф образац у `filter`-нормалној форми.

Лема 3.23 *Нека је D RDF скуп података и G граф у оквиру D. За било који граф образац gp постоји граф образац gp' у `filter`-нормалној форми такав да важи:*

$$[\![gp]\!]_G^D = [\![gp']\!]_G^D.$$

Доказ. Лема је доказана индукцијом по граф обрасцу `gp`.

`gp` је `tp`

По дефиницији 3.31, `gp` је у `filter`-нормалној форми.

`gp` је $\texttt{gp}_1.\texttt{gp}_2$

По дефиницији 3.10, важи

$$[\![\texttt{gp}]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

По индуктивној хипотези примењеној на граф обрасце $\texttt{gp}_1$ и $\texttt{gp}_2$ постоје граф обрасци $\texttt{gp}'_1$ и $\texttt{gp}'_2$ у `filter`-нормалној форми такви да важи

$$[\![\texttt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \text{ и}$$

$$[\![\texttt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

Случај 1: $\texttt{gp}'_1$ и $\texttt{gp}'_2$ су у форми (i), (ii) или (iii).

Нека $\texttt{gp}'$ означава граф образац $\texttt{gp}'_1.\texttt{gp}'_2$. Тада, по дефиницији 3.10, важи

$$[\![\texttt{gp}']\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}},$$

тј.

$$[\![\texttt{gp}']\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

Тада важи $[\![\texttt{gp}]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}']\!]^{\mathrm{D}}_{\mathrm{G}}$ и по дефиницији 3.31, $\texttt{gp}'$ је у `filter`-нормалној форми.

Случај 2: $\texttt{gp}'_1$ је у форми (iv), тј. $\texttt{gp}'_1 = \texttt{gp}''_1$ `filter` $\texttt{R}_1$, док је $\texttt{gp}'_2$ у форми (i), (ii) или (iii).

Тада важи

$$[\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}''_1 \texttt{ filter } \texttt{R}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}},$$

тј. по дефиницији 3.10,

$$[\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} = \{\mu \in [\![\texttt{gp}''_1]\!]^{\mathrm{D}}_{\mathrm{G}} \mid \mu \Vdash \texttt{R}_1\} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

По дефиницији 3.9, ако за пресликавање $\mu \in [\![\texttt{gp}''_1]\!]^{\mathrm{D}}_{\mathrm{G}}$ важи $\mu \Vdash \texttt{R}_1$, тада за његово проширење $\mu \cup \mu_2$, где $\mu_2 \in [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}}$ и $\mu_2 \simeq \mu$, важи $\mu \cup \mu_2 \Vdash \texttt{R}_1$. Дакле, важи:

$$[\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} = \{\mu \cup \mu_2 \in [\![\texttt{gp}''_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} \mid \mu \cup \mu_2 \Vdash \texttt{R}_1\},$$

тј. по дефиницији 3.10,

$$[\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![(\texttt{gp}''_1\texttt{ . }\texttt{gp}'_2) \texttt{ filter } \texttt{R}_1]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

По дефиницији 3.31, $(\texttt{gp}''_1\texttt{ . }\texttt{gp}'_2)$ `filter` $\texttt{R}_1$ је у `filter`-нормалној форми, и важи

$$[\![\texttt{gp}]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![(\texttt{gp}''_1\texttt{ . }\texttt{gp}'_2) \texttt{ filter } \texttt{R}_1]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

Случај 3: $\texttt{gp}'_2$ је у форми (iv), док је $\texttt{gp}'_1$ у форми (i), (ii) или (iii).

Овај случај своди се на претходни због комутативности SPARQL оператора `.` [148], тј. због

$$[\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} = [\![\texttt{gp}'_2]\!]^{\mathrm{D}}_{\mathrm{G}} \bowtie [\![\texttt{gp}'_1]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

Случај 4: $\mathtt{gp}'_1$ и $\mathtt{gp}'_2$ су у форми (iv), тј. у форми $\mathtt{gp}''_1$ `filter` $\mathtt{R}_1$ и $\mathtt{gp}''_2$ `filter` $\mathtt{R}_2$, редом.

Примењујући конструкцију нормалне форме из случајева 2 и 3, добија се:

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![(\mathtt{gp}''_1 \mathtt{.} \mathtt{gp}''_2)\ \mathtt{filter}\ \mathtt{R}_1\ \mathtt{filter}\ \mathtt{R}_2]\!]^{\mathtt{D}}_{\mathtt{G}},$$

тј. по дефиницији 3.9, важи

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![(\mathtt{gp}''_1 \mathtt{.} \mathtt{gp}''_2)\ \mathtt{filter}\ \mathtt{R}_1\mathtt{\&\&}\mathtt{R}_2]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

По дефиницији 3.31, $(\mathtt{gp}''_1 \mathtt{.} \mathtt{gp}''_2)$ `filter` $\mathtt{R}_1\mathtt{\&\&}\mathtt{R}_2$ је у `filter`-нормалној форми.

Може се приметити да све променљиве које се појављују у `filter` клаузули (случајеви 2, 3 и 4) такође се појављују и у граф обрасцу испред. Ово је последица истог својства које важи за иницијалне граф обрасце, па га ова трансформација чува.

$\mathtt{gp}$ је $\mathtt{gp}_1$ `filter` $\mathtt{R}$

По дефиницији 3.10, важи

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = \{\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} \mid \mu \Vdash \mathtt{R}\}.$$

По индуктивној хипотези примењеној на граф образац $\mathtt{gp}_1$, постоји граф образац $\mathtt{gp}'_1$ у `filter`-нормалној форми такав да важи

$$[\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}'_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Тада важи

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = \{\mu \in [\![\mathtt{gp}'_1]\!]^{\mathtt{D}}_{\mathtt{G}} \mid \mu \Vdash \mathtt{R}\},$$

тј. по дефиницији 3.10,

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}'_1\ \mathtt{filter}\ \mathtt{R}]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Ако је граф образац $\mathtt{gp}'_1$ у форми (i), (ii) или (iii), нека $\mathtt{gp}'$ означава $\mathtt{gp}'_1$ `filter` $\mathtt{R}$. Тада важи $[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}']\!]^{\mathtt{D}}_{\mathtt{G}}$, и по дефиницији 3.31, $\mathtt{gp}'$ је у `filter`-нормалној форми.
Ако је $\mathtt{gp}'_1$ у форми (iv), тј. $\mathtt{gp}''_1$ `filter` $\mathtt{R}_1$, нека $\mathtt{gp}'$ означава $\mathtt{gp}''_1$ `filter` $\mathtt{R}\mathtt{\&\&}\mathtt{R}_1$. По дефиницији 3.9, важи

$$[\![\mathtt{gp}''_1\ \mathtt{filter}\ \mathtt{R}_1\ \mathtt{filter}\ \mathtt{R}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}''_1\ \mathtt{filter}\ \mathtt{R}\mathtt{\&\&}\mathtt{R}_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Даље, важи и $[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}']\!]^{\mathtt{D}}_{\mathtt{G}}$, а по дефиницији 3.31, $\mathtt{gp}'$ је у `filter`-нормалној форми.

Може се приметити да такође важи

$$\mathtt{var}(\mathtt{R}\mathtt{\&\&}\mathtt{R}_1) \subseteq \mathtt{var}(\mathtt{gp}''_1),$$

тј. ово својство је очувано и у трансформисаном граф обрасцу.

$\mathtt{gp}$ је $\{\mathtt{gp}_1\}$

По дефиницији 3.10, важи

$$[\![\{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

По индуктивној хипотези примењеној на граф образац $\mathtt{gp}_1$, постоји граф образац $\mathtt{gp}'_1$ у `filter`-нормалној форми такав да важи

$$[\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}'_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Тада, по дефиницији 3.10,

$$[\![\mathtt{gp}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{gp}'_1]\!]^{\mathtt{D}}_{\mathtt{G}},$$

где је $\mathtt{gp}'_1$ у `filter`-нормалној форми.

$\mathtt{gp}$ је $\{\mathtt{Q}_{\mathtt{sq}}\}$

По индуктивној хипотези примењеној на $\mathtt{qpat}_{\mathtt{sq}}$, постоји граф образац $\mathtt{qpat}'_{\mathtt{sq}}$ у `filter`-нормалној форми такав да важи

$$[\![\mathtt{qpat}_{\mathtt{sq}}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\mathtt{qpat}'_{\mathtt{sq}}]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Нека је $\mathtt{Q}'_{\mathtt{sq}}$ упит добијен од упита $\mathtt{Q}_{\mathtt{sq}}$ заменом упит обрасца $\mathtt{qpat}_{\mathtt{sq}}$ обрасцем $\mathtt{qpat}'_{\mathtt{sq}}$. Тада, по дефиницији 3.11, важи

$$[\![\mathtt{Q}_{\mathtt{sq}}]\!]^{\mathtt{D}} = [\![\mathtt{Q}'_{\mathtt{sq}}]\!]^{\mathtt{D}},$$

тј. по дефиницији 3.10(п),

$$[\![\mathtt{qpat}]\!]^{\mathtt{D}}_{\mathtt{G}} = [\![\{\mathtt{Q}'_{\mathtt{sq}}\}]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

По дефиницији 3.31, $\{\mathtt{Q}'_{\mathtt{sq}}\}$ је у `filter`-нормалној форми.

□

Пример 3.17 На слици 3.10 (лево) дат је граф образац $\mathtt{gp}$ који не задовољава `filter`-нормалну форму, и њему еквивалентан граф образац $\mathtt{gp}'$ у `filter`-нормалној форми (десно).

Граф образац $\mathtt{gp}$

```
1 ?x :name         ?n
2 filter (?n = "Tim") .
3 ?x :takesCourse ?y
4 filter (?y = :course04)
```

Граф образац $\mathtt{gp}'$

```
1 {
2   ?x :name         ?n .
3   ?x :takesCourse ?y
4 }
5 filter (?n = "Tim" && ?y = :course04)
```

Слика 3.10: Свођење граф обрасца $\mathtt{gp}$ на еквивалентан образац $\mathtt{gp}'$ у `filter`-нормалној форми

Лема 3.24 *Нека је* D RDF *скуп података и* G *граф у оквиру* D*. За било који граф образац* gp *који садржи подупит* $\{Q_{sq}\}$*, постоји граф образац* gp' *такав да важи*

$$[\![gp]\!]^{D}_{G} = [\![gp']\!]^{D}_{G}$$

и gp' *је у једној од следећих форми:*

- $\{Q'_{sq}\}$,
- $\{Q'_{sq}\}$ `filter` R,
- gp_1 `.` $\{Q'_{sq}\}$,
- gp_1 `.` $\{Q'_{sq}\}$ `filter` R ,

где је gp_1 *у* `filter`*-нормалној форми (i), (ii) или (iii), и* $\{Q'_{sq}\}$ *је у* `filter`*-нормалној форми.*

Доказ. Доказ ове леме је директна последица дефиниције 3.31, леме 3.23 и асоцијативности и комутативности SPARQL оператора `.` [148]. □

Неке променљиве у упиту могу се преименовати, при чему се евалуација трансформисаног упита не мења, што је тврђење следеће леме.

Лема 3.25 *Нека је* $\mathbb{D}$ RDF *скуп података. Нека је* Q *упит, а* $qpat$ *и* $\overline{dv}$ *његов упит образац и његов скуп пројектованих променљивих, редом. Нека је* $qpat'$ *граф образац добијен од* $qpat$ *преименовањем свих променљивих из скупа* $var(qpat) \setminus \overline{dv}$*, уводећи потпуно нове, до тада некоришћене променљиве. Нека је* Q' *упит добијен од* Q *заменом његовог упит обрасца* $qpat$ *упит обрасцем* $qpat'$*. Тада важи:*

$$[\![Q]\!]^{\mathbb{D}} = [\![Q']\!]^{\mathbb{D}}.$$

Доказ. По дефиницији 3.11, важи

$$[\![Q]\!]^{\mathbb{D}} = \Pi_{\overline{\mathrm{dv}}}([\![\mathrm{qpat}]\!]^{\mathrm{D}}_{df(\mathrm{D})}),$$

тј. по дефиницији 3.6,

$$[\![Q]\!]^{\mathbb{D}} = \{\mu_{\overline{\mathrm{dv}}} \mid \mu \in [\![\mathrm{qpat}]\!]^{\mathrm{D}}_{df(\mathrm{D})}\}.$$

Због конструкције обрасца qpat', важи

$$\{\mu_{\overline{\mathrm{dv}}} \mid \mu \in [\![\mathrm{qpat}]\!]^{\mathrm{D}}_{df(\mathrm{D})}\} = \{\mu'_{\overline{\mathrm{dv}}} \mid \mu' \in [\![\mathrm{qpat}']\!]^{\mathrm{D}}_{df(\mathrm{D})}\},$$

јер, по дефиницији 3.5, сва пресликавања из скупа $[\![\mathrm{qpat}]\!]^{\mathrm{D}}_{df(\mathrm{D})}$ и из скупа $[\![\mathrm{qpat}']\!]^{\mathrm{D}}_{df(\mathrm{D})}$ се поклапају на променљивама из скупа $\overline{\mathrm{dv}}$ (у исто време она нису дефинисана на њима, или су дефинисана и имају исте вредности). Зато важи

$$[\![Q]\!]^{\mathbb{D}} = \{\mu'_{\overline{\mathrm{dv}}} \mid \mu' \in [\![\mathrm{qpat}']\!]^{\mathrm{D}}_{df(\mathrm{D})}\}.$$

По дефиницији 3.6 важи и

$$[\![Q]\!]^{\mathbb{D}} = \Pi_{\overline{\mathrm{dv}}}([\![\mathrm{qpat}']\!]^{\mathrm{D}}_{df(\mathrm{D})}).$$

По дефиницији 3.11 и конструкцији упита Q', важи

$$[\![Q]\!]^{\mathbb{D}} = [\![Q']\!]^{\mathbb{D}}.$$

□

Пример 3.18 У упиту Q_1 са слике 3.9, променљива `?z` из подупита може се преименовати у `?z1`, при чему се евалуација упита Q_1 неће променити.

Било који упит $\mathtt{Q}$ који садржи подупит $\mathtt{Q}_{\mathtt{sq}}$ може се свести на еквивалентан упит $\mathtt{Q}'$ који не садржи подупите. Конструкција таквог упита $\mathtt{Q}'$ дата је у доказу следеће леме.

Лема 3.26 *Нека је* $\mathbb{D}$ RDF *скуп података. Нека је* Q *упит чији упит образац садржи подупит* $\{Q_{sq}\}$*. Тада постоји граф образац* gp_{sq} *који није подупут и који, ако се користи у упиту* Q *уместо* $\{Q_{sq}\}$*, формира нови упит* Q*' такав да важи*

$$[\![Q]\!]^{\mathbb{D}} = [\![Q']\!]^{\mathbb{D}}.$$

Доказ. По леми 3.24, постоји граф образац у форми дефинисаној том лемом који је еквивалентан полазном упит обрасцу упита $\mathtt{Q}$. Зато се може подразумевати да је упит образац $\mathtt{qpat}$ упита $\mathtt{Q}$ већ у таквој форми. У наставку, доказан је случај када је

$$\mathtt{qpat} = \mathtt{gp}_1.\{\mathtt{Q}_{\mathtt{sq}}\} \ \ \mathtt{filter\ R}.$$

Остала три случаја су простија, и могу се доказати на сличан начин. У подупиту $\mathtt{Q}_{\mathtt{sq}}$ упит обрасца $\mathtt{qpat}$, могу се преименовати све промењљиве које нису пројектоване ($\mathtt{var}(\mathtt{qpat}_{\mathtt{sq}}) \setminus \overline{\mathtt{dv}_{\mathtt{sq}}}$), уводећи потпуно нове промењљиве које се не појављују у спољашњем упиту $\mathtt{Q}$ (нигде осим у подупиту $\mathtt{Q}_{\mathtt{sq}}$). Овај корак је могућ јер је скуп промењљивих $\mathtt{V}$ пребројив. После овог преименовања, по леми 3.25, евалуација подупита $\mathtt{Q}'_{\mathtt{sq}}$ који садржи новоуведене промењљиве није се променила у поређењу са подупитом $\mathtt{Q}_{\mathtt{sq}}$. Нека $\mathtt{qpat}'_{\mathtt{sq}}$ означава упит образац подупита $\mathtt{Q}'_{\mathtt{sq}}$. Скуп промењљивих у спољашњем упиту $\mathtt{Q}$ садржи пројектоване промењљиве $\overline{\mathtt{dv}}$ и промењљиве граф обрасца $\mathtt{gp}_1$, тј. $\mathtt{var}(\mathtt{gp}_1)$. Зато важи:

$$\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap (\mathtt{var}(\mathtt{gp}_1) \cup \overline{\mathtt{dv}}) \subseteq \overline{\mathtt{dv}_{\mathtt{sq}}}$$

тј.

$$\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \mathtt{var}(\mathtt{gp}_1) \subseteq \overline{\mathtt{dv}_{\mathtt{sq}}} \tag{3.11}$$

$$\text{и } \mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}} \subseteq \overline{\mathtt{dv}_{\mathtt{sq}}} \tag{3.12}$$

Може се приметити да такође важи и $\mathtt{var}(\mathtt{R}) \subseteq \mathtt{var}(\mathtt{gp}_1) \cup \mathtt{var}(\{\mathtt{Q}_{\mathtt{sq}}\})$, тј. по дефиницији 3.7(п),

$$\mathtt{var}(\mathtt{R}) \subseteq \mathtt{var}(\mathtt{gp}_1) \cup \overline{\mathtt{dv}_{\mathtt{sq}}}. \tag{3.13}$$

Нека је $\mathtt{qpat}'$ упит образац који се добија када се у упит обрасцу $\mathtt{qpat}$ подупит $\mathtt{Q}'_{\mathtt{sq}}$ замени обрасцем $\{\mathtt{qpat}'_{\mathtt{sq}}\}$. Нека је $\mathtt{Q}'$ упит који се добија када се у упиту $\mathtt{Q}$ његов упит образац $\mathtt{qpat}$ замени обрасцем $\mathtt{qpat}$'. Докажимо да важи

$$[\![Q]\!]^{\mathbb{D}} = [\![Q']\!]^{\mathbb{D}}.$$

($\subseteq$) Нека је μ пресликавање за које важи $\mu \in [\![Q]\!]^{\mathbb{D}}$. По дефиницији 3.11, постоји пресликавање μ^i, такво да је

$$\mu = \mu^i_{\overline{\mathtt{dv}}} \quad \text{и} \quad \mu^i \in [\![\mathtt{qpat}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

Зато, по дефиницијама 3.10 и 3.10(п), важи

$$\mu^i \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \bowtie [\![\mathtt{Q}'_{\mathtt{sq}}]\!]^{\mathtt{D}} \quad \text{и} \quad \mu^i \Vdash \mathtt{R}.$$

По дефиницији 3.4, постоје компатибилна пресликавања μ_1 и μ^{ii} ($\mu_1 \simeq \mu^{ii}$), таква да важи

$$\mu^i = \mu_1 \cup \mu^{ii} \quad \text{и} \quad \mu_1 \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \quad \text{и} \quad \mu^{ii} \in [\![\mathtt{Q}'_{\mathtt{sq}}]\!]^{\mathtt{D}}.$$

По дефиницији 3.11, постоји пресликавање μ^{iii} такво да је

$$\mu^{ii} = \mu^{iii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}} \quad \text{и} \quad \mu^{iii} \in [\![\mathtt{qpat}'_{\mathtt{sq}}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По дефиницији 3.3, из $\mu_1 \simeq \mu^{ii}$, важи $\mu_1 \simeq \mu^{iii}$, јер не постоји променљива из скупа $\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}})$ а ван скупа $\overline{\mathtt{dv}_{\mathtt{sq}}}$ која се може појавити у $\mathtt{var}(\mathtt{gp}_1)$ због (3.11). Нека μ^{iv} означава пресликавање $\mu_1 \cup \mu^{iii}$. По дефиницији 3.4, важи

$$\mu^{iv} \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \bowtie [\![\mathtt{qpat}'_{\mathtt{sq}}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По дефиницији 3.5, како је $\mu^{iv} = \mu_1 \cup \mu^{iii}$, $\mu^i = \mu_1 \cup \mu^{ii}$ и $\mu^{iii} \succeq \mu^{ii}$, важи и $\mu^{iv} \succeq \mu^i$. Тада, како је μ^{iv} проширење пресликавања μ^i, из $\mu^i \Vdash \mathtt{R}$, следи:

$$\mu^{iv} \Vdash \mathtt{R}.$$

Тада, по дефиницији 3.10, важи

$$\mu^{iv} \in [\![\mathtt{qpat}']\!]^{\mathtt{D}}_{df(\mathtt{D})},$$

тј. по дефиницији 3.11,

$$\mu^{iv}_{\overline{\mathtt{dv}}} \in [\![\mathtt{Q}']\!]^{\mathbb{D}}.$$

Докажимо да је $\mu^{iv}_{\overline{\mathtt{dv}}} = \mu$. По дефиницији 3.5, из $\mu^{iv} \succeq \mu^i$, следи $\mu^{iv}_{\overline{\mathtt{dv}}} \succeq \mu^i_{\overline{\mathtt{dv}}}$, тј. $\mu^{iv}_{\overline{\mathtt{dv}}} \succeq \mu$. Такође, важи и:

$$\begin{aligned}
dom(\mu) \quad & = \\
\text{(по дефиницији 3.5)} \quad & = dom(\mu^i) \cap \overline{\mathtt{dv}} \\
\text{(јер је } \mu^i = \mu_1 \cup \mu^{ii}\text{)} \quad & = (dom(\mu_1) \cup dom(\mu^{ii})) \cap \overline{\mathtt{dv}} \\
\text{(по дистрибутивности)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{ii}) \cap \overline{\mathtt{dv}}) \\
\text{(по дефиницији 3.5)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{iii}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}} \cap \overline{\mathtt{dv}}) \\
\text{(по леми 3.1)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}} \cap \overline{\mathtt{dv}}) \\
\text{(из (3.12))} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}}) \\
\text{(по леми 3.1)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{iii}) \cap \overline{\mathtt{dv}}) \\
\text{(по дистрибутивности)} \quad & = (dom(\mu_1) \cup dom(\mu^{iii})) \cap \overline{\mathtt{dv}} \\
\text{(јер је } \mu^{iv} = \mu_1 \cup \mu^{iii}\text{)} \quad & = dom(\mu^{iv}) \cap \overline{\mathtt{dv}} \\
\text{(по дефиницији 3.5)} \quad & = dom(\mu^{iv}_{\overline{\mathtt{dv}}})
\end{aligned}$$

Дакле, важи $\mu^{iv}_{\overline{\mathtt{dv}}} = \mu$, па и

$$\mu \in [\![\mathtt{Q}']\!]^{\mathbb{D}}.$$

($\supseteq$) Нека је μ пресликавање за које важи $\mu \in [\![\mathtt{Q}']\!]^{\mathbb{D}}$. По дефиницији 3.11, постоји пресликавање μ^i, такво да је

$$\mu = \mu^i_{\overline{\mathtt{dv}}} \quad \text{и} \quad \mu^i \in [\![\mathtt{qpat}']\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

Зато, по дефиницији 3.10, важи

$$\mu^i \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \bowtie [\![\mathtt{qpat}'_{\mathtt{sq}}]\!]^{\mathtt{D}}_{df(\mathtt{D})} \quad \text{и} \quad \mu^i \Vdash \mathtt{R}.$$

По дефиницији 3.4, постоје компатибилна пресликавања μ_1 и μ^{ii} ($\mu_1 \simeq \mu^{ii}$), таква да је

$$\mu^i = \mu_1 \cup \mu^{ii} \quad \text{и} \quad \mu_1 \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \quad \text{и} \quad \mu^{ii} \in [\![\mathtt{qpat}'_{\mathtt{sq}}]\!]^{\mathtt{D}}_{df(\mathtt{D})}.$$

По дефиницији 3.11, важи

$$\mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}} \in [\![\mathtt{Q}'_{\mathtt{sq}}]\!]^{\mathtt{D}}.$$

По дефиницијама 3.3 и 3.5, како је $\mu_1 \simeq \mu^{ii}$, важи $\mu_1 \simeq \mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}}$. Нека μ^{iii} означава пресликавање $\mu_1 \cup \mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}}$. По дефиницији 3.4, важи

$$\mu^{iii} \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{df(\mathtt{D})} \bowtie [\![\mathtt{Q}'_{\mathtt{sq}}]\!]^{\mathtt{D}}.$$

По дефиницији 3.5, важи $\mu_1 \cup \mu^{ii} \succeq \mu_1 \cup \mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}}$, тј. $\mu^i \succeq \mu^{iii}$. Може се приметити да је $dom(\mu^i) = dom(\mu_1) \cup dom(\mu^{ii})$ и $dom(\mu^{iii}) = dom(\mu_1) \cup (dom(\mu^{ii}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}})$, тј. по леми 3.1,

$$dom(\mu^i) = \mathtt{var}(\mathtt{gp}_1) \cup \mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}),$$

$$dom(\mu^{iii}) = \mathtt{var}(\mathtt{gp}_1) \cup (\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}}).$$

Тада важи

$$dom(\mu^i) \setminus dom(\mu^{iii}) = \mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \setminus \overline{\mathtt{dv}_{\mathtt{sq}}}.$$

Дакле, све променљиве из скупа $dom(\mu^i) \setminus dom(\mu^{iii})$ су преименоване, и не могу припадати скупу $\mathtt{var}(\mathtt{R})$, због (3.13). Тада, из $\mu^i \Vdash \mathtt{R}$ следи

$$\mu^{iii} \Vdash \mathtt{R}.$$

Даље, по дефиницији 3.11, важи

$$\mu^{iii}_{\overline{\mathtt{dv}}} \in [\![\mathtt{Q}]\!]^{\mathbb{D}}.$$

Докажимо да је $\mu^{iii}_{\overline{\mathtt{dv}}} = \mu$. По дефиницији 3.5, из $\mu^{iii} \preceq \mu^i$, следи $\mu^{iii}_{\overline{\mathtt{dv}}} \preceq \mu^i_{\overline{\mathtt{dv}}}$, тј. $\mu^{iii}_{\overline{\mathtt{dv}}} \preceq \mu$. Такође, важи и:

$$\begin{aligned}
dom(\mu) \quad & = \\
\text{(по дефиницији 3.5)} \quad & = dom(\mu^i) \cap \overline{\mathtt{dv}} \\
\text{(јер је } \mu^i = \mu_1 \cup \mu^{ii}\text{)} \quad & = (dom(\mu_1) \cup dom(\mu^{ii})) \cap \overline{\mathtt{dv}} \\
\text{(по дистрибутивности)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{ii}) \cap \overline{\mathtt{dv}}) \\
\text{(по леми 3.1)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}}) \\
\text{(из (3.12))} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (\mathtt{var}(\mathtt{qpat}'_{\mathtt{sq}}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}} \cap \overline{\mathtt{dv}}) \\
\text{(по леми 3.1)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{ii}) \cap \overline{\mathtt{dv}_{\mathtt{sq}}} \cap \overline{\mathtt{dv}}) \\
\text{(по дефиницији 3.11)} \quad & = (dom(\mu_1) \cap \overline{\mathtt{dv}}) \cup (dom(\mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}}) \cap \overline{\mathtt{dv}}) \\
\text{(по дистрибутивности)} \quad & = (dom(\mu_1) \cup dom(\mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}})) \cap \overline{\mathtt{dv}} \\
\text{(јер је } \mu^{iii} = \mu_1 \cup \mu^{ii}_{\overline{\mathtt{dv}_{\mathtt{sq}}}}\text{)} \quad & = dom(\mu^{iii}) \cap \overline{\mathtt{dv}} \\
\text{(по дефиницији 3.5)} \quad & = dom(\mu^{iii}_{\overline{\mathtt{dv}}})
\end{aligned}$$

Дакле, важи $\mu^{iii}_{\overline{\mathtt{dv}}} = \mu$, па и

$$\mu \in [\![\mathtt{Q}]\!]^{\mathbb{D}}.$$

□

Пример 3.19 Упит који се добије применом трансформације из доказа леме 3.26 на упит Q_1 са слике 3.9 је управо упит Q_2 са исте слике.

Ако упит Q садржи више од једног подупита, онда се елиминација подупита примењује почевши од најугнежденијег подупита. На тај начин, проблем садржаности упита са подупитима своди се на исти проблем који разматра упите без подупита, а за који је већ показани приступ и сагласан и потпун.

Може се приметити да неодлучивост проблема садржаности упита ако упит Q_2 садржи пројекције проузрокује разматрање само случајева где подупити нису присутни у упиту Q_2 или, ако су присутни, онда су све променљиве из подупита пројектоване ка упиту Q_2.

3.4 Моделовање упита са оператором `union`

Граматика приказана на слици 3.1 садржи основне SPARQL операторе. Један врло важан SPARQL оператор је оператор `union` који проширује граматику језика на начин приказан на слици 3.11. Његовим коришћењем омогућава се комбиновање различитих граф образаца где се један од два могућа граф обрасца примењује, тј. семантика оператора `union` прецизирана је у следећој дефиницији која представља проширење дефиниције 3.10.

```
GPattern  ::= ...
            | GPattern union GPattern
```

Слика 3.11: Додавање оператора `union` граматици са слике 3.1

Дефиниција 3.10(у) (Евалуација граф обрасца) ... Евалуација граф обрасца над графом G у скупу података D, у ознаци $[\![\cdot]\!]^{\mathtt{D}}_{\mathtt{G}}$, дефинисана је рекурзивно са:

$$\ldots := \ldots$$
$$[\![\mathtt{gp}_1 \texttt{ union } \mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}} := [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} \cup [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$$

Пример 3.20 На слици 3.12 (лево), дат је упит Q_1 чији упит образац садржи оператор `union`. Упит враћа студенте основних и мастер студија.

Сваки граф образац gp који садржи оператор `union` може се свести на еквивалентан образац у следећој форми:

$$\mathtt{gp}^1 \texttt{ union } \ldots \texttt{ union } \mathtt{gp}^n,$$

која се зове *проста нормална форма* (енгл. *simple normal form*) граф обрасца, где су сви обрасци $\mathtt{gp}^i (1 \leq i \leq n)$ `union`-слободни граф обрасци [148]. Зато, применом ове трансформације добија се граф образац код ког се оператор `union` може појавити само на највишем нивоу, тј. ван домета свих осталих оператора.

Упит Q_1

```
select ?x
where {
  ?x a :UndergradStudent
  union
  ?x a :MasterStudent
}
```

Упит Q_1^1

```
select ?x
where {
  ?x a :UndergradStudent
}
```

Упит Q_1^2

```
select ?x
where {
  ?x a :MasterStudent
}
```

Слика 3.12: Пример SPARQL упита са оператором `union` (лево), и њему одговарајући упити без оператора `union` (десно)

Лема 3.27 *За упите* $Q_1 = \texttt{select } \overline{dv} \texttt{ where } \{qpat_1\}$ *и* $Q_2 = \texttt{select * where } \{qpat_2\}$ *редом, где су упит обрасци* $qpat_1$ *и* $qpat_2$ *у простој нормалној форми састављени од граф образаца* gp_1^i $(1 \leq i \leq m)$ *и* gp_2^j $(1 \leq j \leq n)$ *редом, нека* Q_1^i $(1 \leq i \leq m)$ *и* Q_2^j $(1 \leq j \leq n)$ *означавају следеће упите:*

$$Q_1^i = \texttt{select } \overline{dv} \texttt{ where } \{gp_1^i\},$$
$$Q_2^j = \texttt{select * where } \{gp_2^j\},$$

где су `from` *и* `from named` *клаузуле упита* Q_1 *и* Q_2 *пропагиране упитима* Q_1^i *и* Q_2^j, *редом. Тада важи:*

$$Q_1 \sqsubseteq Q_2$$
ако и само ако
за свако i, $1 \leq i \leq m$
постоји j, $1 \leq j \leq n$,
такво да је $Q_1^i \sqsubseteq Q_2^j$.

Доказ. ($\Longrightarrow$) Да би овај смер леме био доказан, за свако $\mathtt{i}$, $\mathtt{i} \in \{1, \ldots, \mathtt{m}\}$, треба да постоји $\mathtt{j}$, $\mathtt{j} \in \{1, \ldots, \mathtt{n}\}$, такво да је $\mathtt{Q}_1^{\mathtt{i}} \sqsubseteq \mathtt{Q}_2^{\mathtt{j}}$. Претпоставимо да

$$\mu \in [\![\mathtt{Q}_1^{\mathtt{i}}]\!]^{\mathbb{D}},$$

за неко $\mathtt{i}$, $\mathtt{i} \in \{1, \ldots, \mathtt{m}\}$, и неки скуп података $\mathbb{D}$. По дефиницијама 3.6 и 3.11, постоји пресликавање μ', такво да је $\mu = \mu'_{\overline{\mathtt{dv}}}$, и $\mu' \in [\![\mathtt{gp}_1^{\mathtt{i}}]\!]^{\mathtt{D}_1}_{df(\mathtt{D}_1)}$, где $\mathtt{D}_1$ означава упитни скуп података одређен `from` и `from named` клаузулама упита $\mathtt{Q}_1^{\mathtt{i}}$, али такође и упита $\mathtt{Q}_1$. По дефиницији 3.10(у), важи

$$\mu' \in [\![\mathtt{gp}_1^1 \texttt{ union } \ldots \texttt{ union } \mathtt{gp}_1^{\mathtt{m}}]\!]^{\mathtt{D}_1}_{df(\mathtt{D}_1)}.$$

По дефиницијама 3.6 и 3.11, важи

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}},$$

али такође, како је $\mathtt{Q}_1 \sqsubseteq \mathtt{Q}_2$, и

$$\mu \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

По дефиницијама 3.6 и 3.11, како не постоје пројекције у упиту $\mathtt{Q}_2$, важи

$$\mu \in [\![\mathtt{gp}_2^1 \texttt{ union ... union } \mathtt{gp}_2^{\mathtt{n}}]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)},$$

где $\mathtt{D}_2$ означава скуп података одређен `from` и `from named` клаузулама упита $\mathtt{Q}_2$, али такође и сваког упита $\mathtt{Q}_2^{\mathtt{j}}$, $\mathtt{j} = 1..\mathtt{n}$. По дефиницији 3.10(у), важи

$$\mu \in [\![\mathtt{gp}_2^{\mathtt{j}}]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)},$$

за неки индекс $\mathtt{j}$, $\mathtt{j} \in \{1, \ldots, \mathtt{n}\}$. По дефиницијама 3.6 и 3.11, како не постоје пројекције у упиту $\mathtt{Q}_2^{\mathtt{j}}$, важи

$$\mu \in [\![\mathtt{Q}_2^{\mathtt{j}}]\!]^{\mathbb{D}}.$$

($\Longleftarrow$) Нека је μ неко пресликавање, и $\mathbb{D}$ неки скуп података. Претпоставимо да важи

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

По дефиницијама 3.6 и 3.11, постоји пресликавање μ', такво да је $\mu = \mu'_{\overline{\mathtt{dv}}}$ и

$$\mu' \in [\![\mathtt{gp}_1^1 \texttt{ union ... union } \mathtt{gp}_1^{\mathtt{m}}]\!]^{\mathtt{D}_1}_{df(\mathtt{D}_1)},$$

где $\mathtt{D}_1$ означава упитни скуп података одређен `from` и `from named` клаузулама упита $\mathtt{Q}_1$, али и сваког упита $\mathtt{Q}_1^{\mathtt{i}}$, $\mathtt{i} = 1..\mathtt{m}$. По дефиницији 3.10(у), постоји индекс $\mathtt{i}$, $\mathtt{i} \in \{1, \ldots, \mathtt{m}\}$, такав да $\mu' \in [\![\mathtt{gp}_1^{\mathtt{i}}]\!]^{\mathtt{D}_1}_{df(\mathtt{D}_1)}$. По дефиницијама 3.6 и 3.11, важи

$$\mu \in [\![\mathtt{Q}_1^{\mathtt{i}}]\!]^{\mathbb{D}}.$$

По претпоставци леме, постоји индекс $\mathtt{j}$, $\mathtt{j} \in \{1, \ldots, \mathtt{n}\}$, такав да

$$\mu \in [\![\mathtt{Q}_2^{\mathtt{j}}]\!]^{\mathbb{D}}.$$

По дефиницијама 3.6 и 3.11, како не постоје пројекције у упиту $\mathtt{Q}_2^{\mathtt{j}}$, важи

$$\mu \in [\![\mathtt{gp}_2^{\mathtt{j}}]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)},$$

где $\mathtt{D}_2$ означава упитни скуп података одређен `from` и `from named` клаузулама упита $\mathtt{Q}_2^{\mathtt{j}}$, али такође и упита $\mathtt{Q}_2$. По дефиницији 3.10(у), важи

$$\mu \in [\![\mathtt{gp}_2^1 \texttt{ union ... union } \mathtt{gp}_2^{\mathtt{n}}]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)},$$

тј. по дефиницијама 3.6 и 3.11,

$$\mu \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

□

Пример 3.21 На слици 3.12, за упит $\mathtt{Q}_1$ (лево) приказани су одговарајући упити $\mathtt{Q}_1^1$ и $\mathtt{Q}_1^2$ (десно), који фигуришу у теореми 3.27.

Пратећи лему 3.27, проблем садржаности упита $\mathtt{Q}_1$ и $\mathtt{Q}_2$ своди се на провере да ли за свако i ($1 \leq \mathtt{i} \leq \mathtt{m}$) постоји j ($1 \leq \mathtt{j} \leq \mathtt{n}$) такво да важи $\mathtt{Q}_1^{\mathtt{i}} \sqsubseteq \mathtt{Q}_2^{\mathtt{j}}$. Сагласност и потпуност предложеног моделовања директна су последица претходне леме.

3.5 Моделовање упита са оператором `optional`

Сви граф обрасци објашњени до сада захтевају да буду потпуно упарени. Међутим, некада је јако корисно имати могућност додавања неке информације пресликавању решења када је информација доступна, али не одбацивати потпуно решење у случају да неки део упит обрасца није пронађен. Оператор `optional` обезбеђује ову могућност. Граматика SPARQL упита проширена је на начин приказан на слици 3.13. Семантика оператора `optional` дата је у следећој дефиницији која проширује дефиницију 3.10.

```
GPattern  ::= ...
            | GPattern optional GPattern
```

Слика 3.13: Додавање оператора `optional` граматици са слике 3.1

Дефиниција 3.10(о) (Евалуација граф обрасца) ... Евалуација граф обрасца над графом G у скупу података D, у ознаци $[\![\cdot]\!]^{\mathtt{D}}_{\mathtt{G}}$, дефинисана је рекурзивно са:

$$\ldots := \ldots$$
$$[\![\mathtt{gp}_1 \ \mathtt{optional} \ \mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}} := [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} \bowtie [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$$

Пример 3.22 На слици 3.14 (лево), дат је упит $\mathtt{Q}_1$ чији упит образац садржи оператор `optional`. Упит враћа студенте основних студија и опционо њихове бројеве телефона (ако постоје).

Граф образац `pat` је *добро дефинисан* (енгл. *well-designed*) [151, 149, 148] ако за сваки његов подобразац `gp` такав да је $\mathtt{gp} = \mathtt{gp}_1 \ \mathtt{optional} \ \mathtt{gp}_2$, све променљиве које се појављују у $\mathtt{gp}_2$, али не и у $\mathtt{gp}_1$, не смеју се појављивати у обрасцу `pat` било где другде осим у $\mathtt{gp}_2$. При разматрању и теоријских аспеката и практичне ефикасности извршавања упита, важно је да граф обрасци буду добро дефинисани [151, 149]. Зато, у даљем тексту разматрају се само упити са добро дефинисаним граф обрасцима.

Оператор `optional` може се свести на унију конјункције и негације, како је доказано у следећој леми. Постоје различити типови негације у језику SPARQL, и онај који је овде од користи означава се `diff` оператором [7]. Иако тај оператор још увек није укључен у стандард језика SPARQL, разматран је у поглављу 3.6.1.

Лема 3.28 *Нека је* D *неки* RDF *скуп података, и* G *граф у оквиру њега. Нека су* gp_1 *и* gp_2 *граф обрасци. Тада важи:*

$$[\![gp_1 \ \ optional \ \ gp_2]\!]^{D}_{G} = [\![\{gp_1 \ . \ gp_2\} \ \ union \ \ \{gp_1 \ diff \ gp_2\}]\!]^{D}_{G}.$$

Доказ. Ово је директна последица дефиниција 3.4, 3.10, 3.10(у), 3.10(о) и 3.10(е1). □

Зато, уместо разматрања упита који у себи садрже граф обрасце повезане оператором `optional`, они се синтаксно замењују унијом њихове конјункције (оператор `.`) и њихове негације (оператор `diff`).

Упит Q_1

```
select ?x ?tel
where {
  x a    :UndergradStudent
  optional
  x :tel ?tel
}
```

Упит Q_2

```
select ?x ?tel
where {
  {
    ?x a    :UndergradStudent .
    ?x :tel ?tel
  }
  union
  {
    ?x a    :UndergradStudent
    diff
    ?x :tel ?tel
  }
}
```

Слика 3.14: Пример SPARQL упита са оператором `optional` (лево), и њему еквивалентан упит који се добије свођењем оператора `optional` на операторе `union`, `.` и `diff` (десно)

Пример 3.23 На слици 3.14 приказано је свођење оператора `optional` у упиту Q_1 на операторе `union`, `.` и `diff` у упиту Q_2.

Како су унија и конјункција граф образаца већ разматране, на овом месту неопходно је додати моделовање и доказати сагласност и потпуност само за оператор `diff`. Како се оператор `diff` може појавити независно од оператора `optional` (иако још увек није укључен у стандард језика), сви аспекти овог оператора покривени су у поглављу 3.6.1. Имајући доказану сагласност и потпуност за упите са конјункцијом (поглавље 3.2), унијом (поглавље 3.4) и оператором `diff` (поглавље 3.6.1), проблем садржаности упита који садрже оператор `optional` сведен је на приступ који је и сагласан и потпун.

3.6 Проширења

У овом поглављу, подржано је моделовање SPARQL упита који садрже неке додатне (комплексније и напредније) SPARQL конструкте. Уводе се нови типови граф образаца, израза и услова, један по један, чувајући сагласност приступа у свим случајевима, и потпуност у неким од њих, тј. случајеви где потпуност није очувана јасно су назначени. Такође, у овом делу демонстрирани су неопходни кораци при подржавању новог језичког конструкта у предложеном приступу.

Омогућавање коришћења додатних екстензија урађена је методом постепеног усавршавања (енгл. *stepwise refinement*), додавањем једног по једног конструкта. Међутим, подршка сваког индивидуалног конструкта независна је од осталих, па се проширења представљена у овом делу могу разматрати појединачно или обједињено, бирајући само оне конструкте од интереса. Структура овог поглавља подељена је у делове (за сваки новододати конструкт по један део), и сваки од њих прати структуру претходних поглавља. За сваки нови SPARQL конструкт, граматика упита мора се проширити. Већина дефиниција датих до сада остају исте, а мањи број њих морају се само проширити. Такође, сва тврђења лема и теорема до сада остају да важе без промена, али неки њихови докази морају се проширити. Зато, када је потребно, одговарајуће тврђење биће реферисано и проширење доказа спроведено. У супротном, ако се лема или теорема не наводи у неком одељку, то значи да она остаје да важи и да нема потребе за изменама.

Табела 3.1: Дефиниције, леме и теореме које важе без промена (горњи део) Неопходна проширења дефиниција, и леме са додатним случајевима који треба размотрити у доказима после додавања нових језичких конструката (доњи део)

Без промене

Било које проширење

Дефиниције:	3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.11, 3.12, 3.13, 3.14, 3.15, 3.17, 3.18, 3.19, 3.21, 3.22, 3.23, 3.24, 3.25, 3.27, 3.28
Леме:	3.2, 3.3, 3.7, 3.8, 3.9, 3.10, 3.11, 3.13, 3.16, 3.17, 3.19, 3.20, 3.21
Теореме:	3.18, 3.22

Промена обавезна

Нови тип израза или услова

Дефиниције:	3.7, 3.8, 3.9, 3.16, 3.20, 3.26, 3.29, 3.30
Леме:	3.4, 3.5, 3.12, 3.14

Нови тип граф образаца

Дефиниције:	3.7, 3.10, 3.20
Леме:	3.1, 3.6, 3.15

Табела 3.1 сумира дефиниције, леме и теореме дате у поглављима 3.1 и 3.2, и неопходне информације о евентуалним променама које се морају спровести при додавању новог језичког конструкта. Овакав начин приказа демонстрира даљу проширивост изнетог приступа. У даљем тексту, проширења дефиниција, лема и теорема исто су нумерисана као и њихови оригинали, али са додатним суфиксом (**e***n*) који означава екстензију и њен редни број.

Може се приметити да додавање новог конструкта не утиче на лему 3.13, иако она представља тврђење о изразима. Разлог томе је тај што она представља директну последицу леме 3.12, тј. при додавању нових типова израза, довољно је проширити доказ леме 3.12. Исто важи и за лему 3.16, као последицу леме 3.15.

3.6.1 Оператор `diff`

Као што је дискутовано у поглављу 3.5, сврха оператора `diff` је свођење оператора `optional` на простије операторе `union`, `.` и `diff`. Оператор `diff` представља негацију граф образаца. У ову сврху, стандард језика SPARQL дефинише оператор `minus`, чије су разлике у односу на оператор `diff` мале и незнатне [7], али ипак постоје. Синтакса оператора `diff` дата је на слици 3.15.

```
GPattern  ::= ...
            | GPattern diff GPattern
```

Слика 3.15: Додавање оператора `diff` граматици са слике 3.1

Пример 3.24 Пример употребе оператора `diff` дат је на слици 3.14 (десно). Његова сврха у овом контексту је проналазак свих студената основних студија за које не постоји податак о броју телефона.

Дефиниције 3.7, 3.10 и 3.20, и докази лема 3.1, 3.6 и 3.15 проширују се да би разматрале и овај нови тип обрасца.

Дефиниција 3.7(e1) (Функција `var`) ... Променљиве које се појављују у граф обрасцу `gp`, у ознаци `var(gp)`, су

$$\texttt{var(gp)} := \begin{cases} \ldots, & \ldots \\ \texttt{var}(\texttt{gp}_1), & \texttt{gp je } \texttt{gp}_1 \texttt{ diff } \texttt{gp}_2 \end{cases}$$

Дефиниција 3.10(e1) (Евалуација граф обрасца) ... Евалуација граф обрасца над графом `G` у скупу података `D`, у ознаци $[\![\cdot]\!]^{\texttt{D}}_{\texttt{G}}$, дефинисана је рекурзивно са:

$$\ldots := \ldots$$
$$[\![\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}} := [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}} \setminus [\![\texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}$$

За овај тип граф обрасца такође се подразумева добра дефинисаност, тј. граф образац pat је *добро дефинисан* ако за сваки његов подобразац gp такав да је

$$\texttt{gp} = \texttt{gp}_1 \texttt{ diff } \texttt{gp}_2,$$

све промењљиве које се појављују у $\texttt{gp}_2$, али не и у $\texttt{gp}_1$, не смеју се појављивати у обрасцу pat било где другде осим у $\texttt{gp}_2$. Може се приметити да граф обрасци који се добију редукцијом добро дефинисаних optional образаца задовољавају овај услов.

Дефиниција 3.20(e1) (Функција σ) Функција σ дефинише се рекурзивно у зависности од првог аргумента, тј. ако је аргумент:

- ...
- **Граф образац**:
  - ...
  - оператор diff, резултат је следећа конјункција:
    $\sigma(\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2) := \sigma(\texttt{gp}_1) \wedge \forall \overline{x}\ \neg\sigma(\texttt{gp}_2)$, где $\overline{x}$ означава све промењљиве из скупа $\mathcal{V}$, које се појављују у $\sigma(\texttt{gp}_2)$, али не у $\sigma(\texttt{gp}_1)$.

Лема 3.1(e1)

Доказ. Доказ леме 3.1 допуњен је додатним случајем индукције.

gp је $\texttt{gp}_1$ diff $\texttt{gp}_2$

По дефиницији 3.10(e1), из $\mu \in [\![\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}$, следи

$$\mu \in [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}} \setminus [\![\texttt{gp}_2]\!]^{\texttt{D}}_{\texttt{G}}.$$

Зато, по дефиницији 3.4, важи $\mu \in [\![\texttt{gp}_1]\!]^{\texttt{D}}_{\texttt{G}}$. По индуктивној хипотези важи

$$dom(\mu) = \texttt{var}(\texttt{gp}_1).$$

Тада, по дефиницији 3.7(e1), важи и

$$dom(\mu) = \texttt{var}(\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2)$$

□

Лема 3.6(e1)

Доказ. Доказ леме 3.6 допуњен је додатним случајем индукције.

gp је $\texttt{gp}_1$ diff $\texttt{gp}_2$

$$\begin{aligned}
& var(\sigma(\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2)) = \\
\text{(по дефиницији 3.20(e1))} \quad & = var(\sigma(\texttt{gp}_1) \wedge \forall \overline{x}\ \neg\sigma(\texttt{gp}_2)) \\
\text{(по дефиницији 3.21)} \quad & = var(\sigma(\texttt{gp}_1)) \\
\text{(по индуктивној хипотези)} \quad & = \sigma(\texttt{var}(\texttt{gp}_1)) \\
\text{(по дефиницији 3.7(e1))} \quad & = \sigma(\texttt{var}(\texttt{gp}_1 \texttt{ diff } \texttt{gp}_2))
\end{aligned}$$

□

Лема 3.15(e1)

Доказ. Доказ леме 3.15 допуњен је додатним случајем индукције.

gp је $\mathtt{gp}_1$ `diff` $\mathtt{gp}_2$

($\Rightarrow$) По дефиницији 3.10(e1), из $\mu \in [\![\mathtt{gp}_1 \ \ \mathtt{diff} \ \ \mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$ следи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}} \setminus [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

По дефиницији 3.4, важи $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}$ и не постоји пресликавање μ_2 такво да је $\mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$ и $\mu \simeq \mu_2$. По индуктивној хипотези, због $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}$, за пресликавање μ и одговарајућу валуацију ν ($\mu \leftrightarrow \nu$), важи

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \text{ и } dom(\nu) = var(\sigma(\mathtt{gp}_1)).$$

Контрапозицијом доказује се да важи $(\mathfrak{D}, \nu) \models \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2)$, тј. претпоставимо да важи

$$(\mathfrak{D}, \nu) \models \exists \overline{x}\ \sigma(\mathtt{gp}_2).$$

Ово значи да се валуација ν може проширити на све промењиве из $var(\sigma(\mathtt{gp}_2))$ (додајући вредности промењивама из $\overline{x}$), и онда направити рестрикцију на само оне промењиве из $var(\sigma(\mathtt{gp}_2))$ (уклањајући промењиве из $var(\sigma(\mathtt{gp}_1))$ које нису у $var(\sigma(\mathtt{gp}_2))$). Нека ν_2 означава такву валуацију. Тада важи

$$(\mathfrak{D}, \nu_2) \models \sigma(\mathtt{gp}_2).$$

Може се приметити да због конструкције важи $\nu_2 \simeq \nu$. Из индуктивне хипотезе примењене на валуацију ν_2 и пресликавање μ_2 такво да је $\mu_2 \leftrightarrow \nu_2$, следи да $\mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$. По леми 3.10, из $\mu \leftrightarrow \nu$, $\mu_2 \leftrightarrow \nu_2$ и $\nu \simeq \nu_2$, следи $\mu \simeq \mu_2$. Ово је контрадикција са непостојањем пресликавања из $[\![\mathtt{gp}_2]\!]^{\mathtt{D}}_{\mathtt{G}}$ компатибилног пресликавању μ. Дакле, важи:

$$(\mathfrak{D}, \nu) \models \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2).$$

($\Leftarrow$) По дефиницији 3.20(e1), из

$$dom(\nu) = var(\sigma(\mathtt{gp}_1 \ \ \mathtt{diff} \ \ \mathtt{gp}_2)), \text{ и}$$

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1 \ \ \mathtt{diff} \ \ \mathtt{gp}_2)$$

следи

$$dom(\nu) = var(\sigma(\mathtt{gp}_1) \wedge \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2)), \text{ и}$$
$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \wedge \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2),$$

тј. по дефиницији 3.27,

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu) \models \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2).$$

По дефиницији 3.21, како $\overline{x}$ означава све промењиве које се појављују у $\sigma(\mathtt{gp}_2)$, а не у $\sigma(\mathtt{gp}_1)$, важи $dom(\nu) = var(\sigma(\mathtt{gp}_1))$. Применом индуктивне хипотезе на валуацију ν и одговарајуће пресликавање μ ($\mu \leftrightarrow \nu$), важи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

Такође треба доказати да не постоји пресликавање μ_2 такво да $\mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}$ и $\mu_2 \simeq \mu$. Ове се доказује контрапозицијом, тј. претпоставком о постојању таквог пресликавања μ_2. Применом индуктивне хипотезе на пресликавање μ_2 и валуацију ν_2 такву да је $\mu_2 \leftrightarrow \nu_2$, следи

$$(\mathfrak{D}, \nu_2) \models \sigma(\mathtt{gp}_2).$$

По леми 3.10, из $\mu_2 \simeq \mu$, $\mu \leftrightarrow \nu$ и $\mu_2 \leftrightarrow \nu_2$, следи $\nu_2 \simeq \nu$. Тада је $\nu \cup \nu_2$ добро дефинисана валуација, која ће бити означавана са ν'. Како је ν' проширење валуације ν_2, из $(\mathfrak{D}, \nu_2) \models \sigma(\mathtt{gp}_2)$, следи

$$(\mathfrak{D}, \nu') \models \sigma(\mathtt{gp}_2).$$

Како је ν' проширење валуације ν, из $(\mathfrak{D}, \nu) \models \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2)$, следи

$$(\mathfrak{D}, \nu') \models \forall \overline{x}\ \neg\sigma(\mathtt{gp}_2).$$

Тада постоји проширење ν'' валуације ν', такво да важи

$$(\mathfrak{D}, \nu'') \models \neg\sigma(\mathtt{gp}_2).$$

Такође, како је ν'' проширење валуације ν', из $(\mathfrak{D}, \nu') \models \sigma(\mathtt{gp}_2)$, следи

$$(\mathfrak{D}, \nu'') \models \sigma(\mathtt{gp}_2).$$

Дакле, $(\mathfrak{D}, \nu'')$ је модел формуле $\sigma(\mathtt{gp}_2)$ и њене негације, што чини контрадикцију. Како важи $\mu \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}}$ и не постоји μ_2 такво да $\mu_2 \in [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}$ и $\mu_2 \simeq \mu$, по дефиницији 3.4 важи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathrm{D}}_{\mathrm{G}} \setminus [\![\mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

Тада, по дефиницији 3.10(e1), важи и

$$\mu \in [\![\mathtt{gp}_1 \quad \mathtt{diff} \quad \mathtt{gp}_2]\!]^{\mathrm{D}}_{\mathrm{G}}.$$

□

3.6.2 Приступ именима графова и рестрикција над њима

SPARQL упит може захтевати претрагу граф образаца по сваком од именованих графова у RDF скупу података и формирати решење које одговарајућим променљивама придружује имена претраживаног графа. Ово се остварује кроз `graph` конструкт. Његово друго својство ограничава претрагу на тачно одређени граф чије је име тачно одређено прослеђеним IRI-јем. На овај начин поставља се активни граф на тако прецизирани граф. Граматика SPARQL упита проширује се на начин приказан на слици 3.16.

```
GPattern  ::= ...
            | graph var '{' GPattern '}'
            | graph iri '{' GPattern '}'
```

Слика 3.16: Додавање конструкта `graph` граматици са слике 3.1

Пример 3.25 На слици 3.17, дати су упити Q_1 и Q_2, чији упит обрасци садрже оператор `graph`. Упит Q_1 враћа студенте основних студија из било ког именованог графа заједно са именом тог графа, док упит Q_2 враћа само студенте из именованог графа чији је IRI `<http://example.org/>`.

Упит Q_1

```
select ?x ?g
where {
  graph ?g {
    ?x a :UndergradStudent
  }
}
```

Упит Q_2

```
select ?x
where {
  graph <http://example.org/> {
    ?x a :UndergradStudent
  }
}
```

Слика 3.17: Пример SPARQL упита са оператором `graph`

Дефиниције 3.7, 3.10 и 3.20, и докази лема 3.1, 3.6 и 3.15 проширују се да би разматрале и овај нови тип обрасца.

Дефиниција 3.7(e2) (Функција `var`) ... Променљиве које се појављују у граф обрасцу `gp`, у ознаци `var(gp)`, су

$$\texttt{var(gp)} := \begin{cases} ..., & ... \\ \texttt{var}(\texttt{gp}_1) \cup \{\texttt{x}\}, & \texttt{gp je graph x } \{\texttt{gp}_1\} \\ \texttt{var}(\texttt{gp}_1), & \texttt{gp je graph i } \{\texttt{gp}_1\} \end{cases}$$

Дефиниција 3.10 проширена је у складу са литературом [5].

Дефиниција 3.10(e2) (Евалуација граф обрасца) ... Нека је `i` IRI, и `x` променљива. Евалуација граф обрасца над графом `G` у скупу података `D`, у ознаци $[\![\cdot]\!]^{\texttt{D}}_{\texttt{G}}$, дефинисана је рекурзивно са:

$$\begin{aligned} ... &:= ... \\ [\![\texttt{graph x } \{\texttt{gp}\}]\!]^{\texttt{D}}_{\texttt{G}} &:= \bigcup_{\texttt{i} \in names(\texttt{D})} \left([\![\texttt{gp}]\!]^{\texttt{D}}_{gr_{\texttt{D}}(\texttt{i})} \bowtie \{\mu_{\texttt{x} \to \texttt{i}}\} \right) \\ [\![\texttt{graph i } \{\texttt{gp}\}]\!]^{\texttt{D}}_{\texttt{G}} &:= [\![\texttt{gp}]\!]^{\texttt{D}}_{gr_{\texttt{D}}(\texttt{i})} \end{aligned}$$

Дефиниција 3.20(e2) (Функција σ) Функција σ дефинише се рекурзивно, у зависности од првог аргумента, тј. ако је аргумент:

- ...
- **Граф образац**:

  Нека скуп I_n садржи све граф IRI константе које одговарају именованим графовима упитног скупа `D` одређеног упитном, тј. $I_n := \{i_{l_1}, \cdots, i_{l_n}\}$, где $i_{l_j} := \sigma(\texttt{i}_{l_j})$,

и IRI-јеви $\mathtt{i}_{l_j}$ су дати `from named` клаузулама упита, или, у случају да нема ни `from named` ни `from` клаузули, скуп I_n је једнак $\sigma_{\mathtt{t}}(names(\mathbb{D}))$. Ако нема присутне `from named` клаузуле, али постоји нека `from` клаузула, овај скуп је празан, тј. $I_n := \varnothing$.

– ...

– граф образац $\mathtt{gp}_1$ који треба бити претражен по свим именованим графовима, резултат је следећа формула:

$$\sigma(\texttt{graph x } \{\mathtt{gp}_1\}) := \begin{cases} \bigvee\limits_{i \in I_n} \left(\sigma^{gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = i \right), & I_n \neq \varnothing \\ \bot, & I_n = \varnothing, \end{cases}$$

– граф образац $\mathtt{gp}_1$ који треба бити претражен по одређеном именованом графу, резултат је следећа формула:

$$\sigma(\texttt{graph i } \{\mathtt{gp}_1\}) := \begin{cases} \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1), & \sigma(\mathtt{i}) \in I_n \\ \bot, & \sigma(\mathtt{i}) \notin I_n \ . \end{cases}$$

Лема 3.1(e2)

Доказ. Доказ леме 3.1 допуњен је додатним случајем индукције.

`gp` је `graph x` $\{\mathtt{gp}_1\}$

По дефиницији 3.10(e2), из $\mu \in [\![\texttt{graph x } \{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$, следи

$$\mu \in \bigcup_{\mathtt{i} \in names(\mathtt{D})} \left([\![\mathtt{gp}]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \bowtie \{\mu_{\mathtt{x} \to \mathtt{i}}\} \right).$$

Зато, по дефиницији 3.4 и индуктивној хипотези, важи

$$dom(\mu) = \mathtt{var}(\mathtt{gp}_1) \cup \{\mathtt{x}\}.$$

Тада, по дефиницији 3.7(e2), важи и

$$dom(\mu) = \mathtt{var}(\texttt{graph x } \{\mathtt{gp}_1\}).$$

`gp` је `graph i` $\{\mathtt{gp}_1\}$

По дефиницији 3.10(e2), из $\mu \in [\![\texttt{graph i } \{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$, следи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})}.$$

Зато, по индуктивној хипотези, важи

$$dom(\mu) = \mathtt{var}(\mathtt{gp}_1).$$

Тада, по дефиницији 3.7(e2), важи

$$dom(\mu) = \mathtt{var}(\texttt{graph i } \{\mathtt{gp}_1\}).$$

□

Лема 3.6(e2)

Доказ. Доказ леме 3.6 допуњен је додатним случајем индукције.

gp је `graph x` $\{\mathtt{gp}_1\}$

Ако је $I_n = \varnothing$, тада не постоји активни граф за претрагу обрасца $\mathtt{gp}_1$. Тада, по дефиницији 3.10(e2), овај случај је еквивалентан случају када се празан граф користи као активни, што не одговара претпоставци леме. Зато, може се претпоставити да је $I_n \neq \varnothing$. Из тога, по дефиницији 3.20(e2), за свако $i \in I_n$, граф $gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))$ је непразан граф. Важи и следеће:

$$
\begin{aligned}
& var(\sigma(\texttt{graph x } \{\mathtt{gp}_1\})) = \\
\text{(по дефиницији 3.20(e2))} \quad & = var(\bigvee_{i \in I_n} (\sigma^{gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = i)) \\
\text{(по дефиницији 3.21)} \quad & = \bigcup_{i \in I_n} (var(\sigma^{gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))}(\mathtt{gp}_1) \cup var(\sigma(\mathtt{x}) = i)) \\
\text{(по индуктивној хипотези)} \quad & = \bigcup_{i \in I_n} (\sigma(\mathtt{var}(\mathtt{gp}_1)) \cup var(\sigma(\mathtt{x}) = i)) \\
\text{(како је } I_n \cap \mathcal{V} = \varnothing) \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1)) \cup \{\sigma(\mathtt{x})\} \\
\text{(по дефиницији 3.18)} \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1) \cup \{\mathtt{x}\}) \\
\text{(по дефиницији 3.7(e2))} \quad & = \sigma(\mathtt{var}(\texttt{graph x } \{\mathtt{gp}_1\}))
\end{aligned}
$$

gp је `graph i` $\{\mathtt{gp}_1\}$

Ако $\sigma(\mathtt{i}) \notin I_n$, по дефиницији 3.20(e2), важи $\mathtt{i} \notin names(\mathtt{D})$, тј. по дефиницији 3.1, $gr_{\mathtt{D}}(\mathtt{i}) = \varnothing$. Тада, по дефиницији 3.20(e2), активни граф који се користи при претрази обрасца $\mathtt{gp}_1$ је празан, што не одговара претпоставци леме. Зато се може претпоставити да $\sigma(\mathtt{i}) \in I_n$. Из тога, по дефиницији 3.20(e2), граф $gr_{\mathtt{D}}(\mathtt{i})$ је непразан. Важи и следеће:

$$
\begin{aligned}
& var(\sigma(\texttt{graph i } \{\mathtt{gp}_1\})) = \\
\text{(по дефиницији 3.20(e2))} \quad & = var(\sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1)) \\
\text{(по индуктивној хипотези)} \quad & = \sigma(\mathtt{var}(\mathtt{gp}_1)) \\
\text{(по дефиницији 3.7(e2))} \quad & = \sigma(\mathtt{var}(\texttt{graph i } \{\mathtt{gp}_1\}))
\end{aligned}
$$

□

Лема 3.15(e2)

Доказ. Доказ леме 3.15 допуњен је додатним случајем индукције.

gp је `graph x` $\{\mathtt{gp}_1\}$

($\Longrightarrow$) По дефиницији 3.10(e2), из $\mu \in [\![\texttt{graph x } \{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$ следи

$$\mu \in \bigcup_{\mathtt{i} \in names(\mathtt{D})} \left([\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \bowtie \{\mu_{\mathtt{x} \to \mathtt{i}}\} \right).$$

Дакле, $I_n \neq \varnothing$ и постоји IRI i из $names(\mathtt{D})$ такав да

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \bowtie \{\mu_{\mathtt{x} \to \mathtt{i}}\}.$$

По дефиницији 3.4, постоји пресликавање μ_1 такво да је

$$\mu = \mu_1 \cup \mu_{\mathtt{x}\to\mathtt{i}},\ \mu_1 \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \text{ и } \mu_1 \simeq \mu_{\mathtt{x}\to\mathtt{i}}.$$

Примењујући индуктивну хипотезу на пресликавање μ_1 и валуацију ν_1 такву да је $\mu_1 \leftrightarrow \nu_1$, важи

$$(\mathfrak{D}, \nu_1) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1).$$

По леми 3.11, из $\mu = \mu_1 \cup \mu_{\mathtt{x}\to\mathtt{i}}$, $\mu \leftrightarrow \nu$, $\mu_1 \leftrightarrow \nu_1$ и $\mu_{\mathtt{x}\to\mathtt{i}} \leftrightarrow \nu_{x\to\mathtt{i}}$, важи $\nu = \nu_1 \cup \nu_{x\to\mathtt{i}}$. Зато, ν је проширење валуације ν_1, па важи и

$$(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1).$$

Из $\mu = \mu_1 \cup \mu_{\mathtt{x}\to\mathtt{i}}$ важи $\mu(\mathtt{x}) = \mathtt{i}$. Како је $\mathtt{i} \in \mathtt{I}$, по дефиницији 3.8, важи $\mathtt{i} = [[\mathtt{i}]]_\mu$, па је $\mu(\mathtt{x}) = [[\mathtt{i}]]_\mu$. По леми 3.13, из тога и из $\mu \leftrightarrow \nu$, важи

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{x}) = \sigma(\mathtt{i}).$$

Тада, по дефиницији 3.27, важи

$$(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = \sigma(\mathtt{i}).$$

По дефиницији 3.20(e2), из $\mathtt{i} \in names(\mathtt{D})$, за $i = (\sigma_{\mathtt{t}})^{-1}(\mathtt{i})$, следи $i \in I_n$. Тада важи

$$(\mathfrak{D}, \nu) \models \bigvee_{i \in I_n} \Big(\sigma^{gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = i\Big),$$

и, по дефиницији 3.20(e2),

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{graph\ x\ \{gp_1\}}).$$

($\Longleftarrow$) Из

$$(\mathfrak{D}, \nu) \models \sigma(\mathtt{graph\ x\ \{gp_1\}}) \text{ и } dom(\nu) = var(\sigma(\mathtt{graph\ x\ \{gp_1\}}))$$

по дефиницији 3.20(e2), важи

$$(\mathfrak{D}, \nu) \models \bigvee_{i \in I_n} \Big(\sigma^{gr_{\mathtt{D}}((\sigma_{\mathtt{t}})^{-1}(i))}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = i\Big) \text{ и } dom(\nu) = var(\sigma(\mathtt{gp}_1)) \cup \{\sigma(\mathtt{x})\},$$

Дакле, постоји константа i из скупа I_n, таква да важи $i = \sigma(\mathtt{i})$ и

$$(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1) \wedge \sigma(\mathtt{x}) = \sigma(\mathtt{i}).$$

Отуд, по дефиницији 3.27, важи

$$(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1) \text{ и } (\mathfrak{D}, \nu) \models \sigma(\mathtt{x}) = \sigma(\mathtt{i}).$$

По леми 3.13, из друге задовољивости и $\mu \leftrightarrow \nu$, важи $\mu(x) = [[\mathtt{i}]]_\mu$, тј. по дефиницији 3.8, важи $\mu(x) = \mathtt{i}$. Разматрају се следећа два случаја:

- Ако $\sigma(\mathtt{x}) \in var(\sigma(\mathtt{gp}_1))$, тј. $dom(\nu) = var(\sigma(\mathtt{gp}_1))$. По индуктивној хипотези, из прве задовољивости и $\mu \leftrightarrow \nu$, важи $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})}$. Тада је $\mu = \mu \cup \mu_{\mathtt{x}\to\mathtt{i}}$.

- Ако $\sigma(\mathtt{x}) \notin var(\sigma(\mathtt{gp}_1))$, нека ν' означава рестрикцију ν на скуп $var(\sigma(\mathtt{gp}_1))$. Тада важи
$$(\mathfrak{D}, \nu') \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1).$$
Даље, за пресликавање μ' такво да је $\mu' \leftrightarrow \nu'$, по индуктивној хипотези, важи $\mu' \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})}$. По дефиницији 3.28, из $\mu \leftrightarrow \nu$, $\mu' \leftrightarrow \nu'$ и $\nu' \preceq \nu$, следи $\mu' \preceq \mu$. Такође важи:

$dom(\mu)$	$=$
(по дефиницији 3.28, јер је $\mu \leftrightarrow \nu$)	$= (\sigma_{\mathtt{t}})^{-1}(dom(\nu))$
(јер је $dom(\nu) = var(\sigma(\mathtt{gp}_1)) \cup \{\sigma(\mathtt{x})\}$)	$= (\sigma_{\mathtt{t}})^{-1}(var(\sigma(\mathtt{gp}_1)) \cup \{\sigma(\mathtt{x})\})$
(по дефиницији 3.18)	$= (\sigma_{\mathtt{t}})^{-1}(var(\sigma(\mathtt{gp}_1))) \cup (\sigma_{\mathtt{t}})^{-1}(\{\sigma(\mathtt{x})\})$
(по леми 3.6)	$= (\sigma_{\mathtt{t}})^{-1}(\sigma(var(\mathtt{gp}_1))) \cup (\sigma_{\mathtt{t}})^{-1}(\{\sigma(\mathtt{x})\})$
(по дефиницији 3.17, 3.20)	$= \mathtt{var}(\mathtt{gp}_1) \cup \{\mathtt{x}\}$

По леми 3.1, како је $\mu' \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})}$, важи $dom(\mu') = \mathtt{var}(\mathtt{gp}_1)$. Тада важи $\mu = \mu' \cup \mu_{\mathtt{x}\to\mathtt{i}}$.

У оба случаја, по дефиницији 3.4, важи

$$\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \bowtie \{\mu_{\mathtt{x}\to\mathtt{i}}\},$$

тј.

$$\mu \in \bigcup_{\mathtt{i} \in names(\mathtt{D})} \left([\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})} \bowtie \{\mu_{\mathtt{x}\to\mathtt{i}}\} \right).$$

Најзад, по дефиницији 3.10(e2), важи

$$\mu \in [\![\texttt{ graph x } \{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}.$$

gp је `graph i {`$\mathtt{gp}_1$`}`

$\mu \in [\![\texttt{ graph i } \{\mathtt{gp}_1\}]\!]^{\mathtt{D}}_{\mathtt{G}}$	акко
(по дефиницији 3.10(e2))	акко $\mu \in [\![\mathtt{gp}_1]\!]^{\mathtt{D}}_{gr_{\mathtt{D}}(\mathtt{i})}$
(по индуктивној хипотези)	акко $(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1))$
(по дефиницији 3.7(e2))	акко $(\mathfrak{D}, \nu) \models \sigma^{gr_{\mathtt{D}}(\mathtt{i})}(\mathtt{gp}_1)$ и $dom(\nu) = var(\sigma(\mathtt{gp}))$
(по дефиницији 3.20(e2))	акко $(\mathfrak{D}, \nu) \models \sigma(\texttt{graph i } \{\mathtt{gp}_1\})$ и $dom(\nu) = var(\sigma(\mathtt{gp}))$

□

3.6.3 Проширења израза и услова

Језик SPARQL подржава широк спектар уграђених функција које се могу користити у изразима и условима. По једна функција из обе класе приказана је у овом поглављу, док се остале могу разматрати на сличан начин. Граматика SPARQL упита проширује се на начин приказан на слици 3.18.

```
Expr     := ...                          Cond     := ...
          | datatype '(' Expr ')'                   | isliteral '(' Expr ')'
```

Слика 3.18: Додавање нових израза и услова граматици са слике 3.1

Пример 3.26 На слици 3.19, дати су упити Q_1 и Q_2, који користе уграђене функције `datatype` и `isliteral`, редом. Упити враћају студенте основних студија и њихова имена, при чему имена морају бити у случају упита Q_1 типа `xsd:string`, а у случају Q_2 литерали.

Упит Q_1

```
1 select ?x ?n
2 where {
3   ?x a      :UndergradStudent .
4   ?x :name ?n
5   filter (datatype(?n) = xsd:string)
6 }
```

Упит Q_2

```
1 select ?x ?n
2 where {
3   ?x a      :UndergradStudent .
4   ?x :name ?n
5   filter (isliteral(?n))
6 }
```

Слика 3.19: Пример SPARQL упита који користе уграђене функције `datatype` и `isliteral`

Дефиниције 3.7, 3.8, 3.9, 3.16, 3.20, 3.26, 3.29 и 3.30, и докази лема 3.4, 3.5, 3.12, и 3.14 проширују се да би обухватили и ове нове типове израза и услова.

Дефиниција 3.7(e3) (Функција *var*) ... Променљиве које се појављују у изразу `E` и услову `R`, у ознаци `var(E)` и `var(R)`, редом, су

$$var(\texttt{E}) := \begin{cases} \ldots & \ldots \\ var(\texttt{E}_1), & \texttt{E} \text{ је } \texttt{datatype(E}_1\texttt{)} \end{cases}$$

$$var(\texttt{R}) := \begin{cases} \ldots & \ldots \\ var(\texttt{E}), & \texttt{R} \text{ је } \texttt{isliteral(E)} \end{cases}$$

У складу са семантиком језика SPARQL [155], потребна је следећа функција да би се дефиниција 3.8 проширила.

Дефиниција 3.32 (Семантика функције dt) Нека је l литерал. Функција $\texttt{dt} : \texttt{L} \to \texttt{I}$ дефинисана је са:

$$\texttt{dt(l)} := \begin{cases} \texttt{i}, & \texttt{l је типизиран литерал са \^{}\^{}i у суфиксу где i} \in \texttt{I} \\ \texttt{xsd:string}, & \texttt{l је нетипизиран литерал.} \end{cases}$$

Дефиниција 3.8(e3) (Нотација $[[\cdot]]_\mu$) ... Вредност израза E, у пресликавању μ, у ознаци $[[\texttt{E}]]_\mu$, је вредност из скупа IBLe, дефинисана са:

$$[[\texttt{datatype(E}_1\texttt{)}]]_\mu := \begin{cases} \texttt{dt}([[\texttt{E}_1]]_\mu), & [[\texttt{E}_1]]_\mu \neq \texttt{err и } [[\texttt{E}_1]]_\mu \in \texttt{L} \\ \texttt{err}, & \text{иначе.} \end{cases}$$

Дефиниција 3.9(e3) (Релација $\Vdash$) ... Пресликавање μ задовољава услов R, у ознаци $\mu \Vdash \texttt{R}$, ако:

$$\mu \Vdash \texttt{R} := \begin{cases} \ldots, & \ldots \\ [[\texttt{E}]]_\mu \neq \texttt{err и } [[\texttt{E}]]_\mu \in \texttt{L}, & \texttt{R je isliteral(E)} \end{cases}$$

Дефиниција 3.16(e3) (Сигнатура $\mathcal{L}$ која одговара упитима $\texttt{Q}_1$ и $\texttt{Q}_2$)
...

- $\mathcal{F}_s := \mathcal{C} \cup \mathcal{F}$ је скуп функцијских симбола, где:
  - ...
  - $\mathcal{F} := \{datatype\}$.
- $\mathcal{P}_s := \mathcal{P} \cup \{\beta_n, \beta_d\}$ је скуп предикатских симбола, где:
  - $\mathcal{P} := \{..., isliteral\}$.
  - ...
- Функција арности дефинисана је са:

$$ar(\alpha) := \begin{cases} ..., & ... \\ 1, & \alpha \in \mathcal{F},\ \alpha \text{ је } datatype \\ 1, & \alpha \in \mathcal{P}_s,\ \alpha \text{ је } isliteral. \end{cases}$$

Дефиниција 3.20(e3) (Функција σ) Функција σ дефинише се рекурзивно у зависности од првог аргумента, тј. ако је аргумент:

- **Израз E**:
  - ...
  - datatype(E), резултат је одговарајућа неинтерпретирана функција: $\sigma(\texttt{datatype(E)}) := datatype(\sigma(\texttt{E}))$

- **Услов R**:
  - ...
  - `isliteral(E)`, резултат је одговарајућа неинтерпретирана функција: $\sigma(\texttt{isliteral(E)}) := \mathit{isliteral}(\sigma(\texttt{E}))$

Дефиниција 3.26(e3) ($\mathcal{L}$-структура $\mathfrak{D}$) ...

- ...
- $\mathcal{I}^{\mathcal{D}}$ је функција која пресликава нелогичке термове сигнатуре на следећи начин:
  - ...
  - Интерпретација $\mathcal{I}^{\mathcal{D}}$ функцијског симбола *datatype* је функција $\texttt{DATATYPE} : \texttt{IBL} \to \texttt{IBLe}$, дефинисана са:

$$\texttt{DATATYPE(t)} := \begin{cases} \texttt{dt(t)}, & \texttt{t} \in \texttt{L} \\ \texttt{err}, & \text{иначе,} \end{cases}$$

  - Интерпретација $\mathcal{I}^{\mathcal{D}}$ предикатског симбила *isliteral* је функција $\texttt{ISLITERAL} : \texttt{IBL} \to \mathit{bool}$, дефинисана са:

$$\texttt{ISLITERAL(t)} = \top \text{ ако и само ако } \texttt{t} \in \texttt{L}\ .$$

Дефиниција 3.29(e3) (Нотација $[[\cdot]]_\nu$) ..., и t_1 је RDF терм из SPARQL сигнатуре $\mathcal{L}$. ...

$$[[t]]_\nu := \begin{cases} \texttt{err}\,, & t \in \mathcal{V} \text{ и } t \notin \mathit{dom}(\nu) \\ \texttt{err}\,, & t \text{ је } \mathit{datatype}(t_1) \text{ и } [[t_1]]_\nu = \texttt{err} \\ \texttt{err}\,, & t \text{ је } \mathit{isliteral}(t_1) \text{ и } [[t_1]]_\nu = \texttt{err} \\ \mathcal{I}_\nu(t)\,, & \text{иначе.} \end{cases}$$

Коришћењем неинтерпретираних функција за моделовање SPARQL уграђених функција, приступ губи потпуност, тј. теорема 3.22 не може се доказати за овај тип проширења. Како се дефиниција 3.30 користи само у контексту ове теореме, њено проширење није наведено.

Лема 3.4(e3)

Доказ. Доказ леме 3.4 допуњен је додатним случајем индукције.

E је $\texttt{datatype(E}_1\texttt{)}$

$$\begin{aligned} & \mathit{var}(\sigma(\texttt{datatype(E}_1\texttt{)})) = \\ \text{(по дефиницији 3.20(e3))} \quad & = \mathit{var}(\mathit{datatype}(\sigma(\texttt{E}_1))) \\ \text{(по дефиницији 3.21)} \quad & = \mathit{var}(\sigma(\texttt{E}_1)) \\ \text{(по индуктивној хипотези)} \quad & = \sigma(\texttt{var}(\texttt{E}_1)) \\ \text{(по дефиницији 3.7(e3))} \quad & = \sigma(\texttt{var}(\texttt{datatype(E}_1\texttt{)})) \end{aligned}$$

□

Лема 3.5(e3)

Доказ. Доказ леме 3.5 допуњен је додатним случајем индукције.

R је `isliteral(E)`

$$
\begin{aligned}
var(\sigma(\texttt{isliteral(E)})) &= \\
\text{(по дефиницији 3.20(e3))} &= var(isliteral(\sigma(\texttt{E}))) \\
\text{(по дефиницији 3.21)} &= var(\sigma(\texttt{E})) \\
\text{(по индуктивној хипотези)} &= \sigma(\texttt{var(E)}) \\
\text{(по дефиницији 3.7(e3))} &= \sigma(\texttt{var(isliteral(E))})
\end{aligned}
$$

□

Лема 3.12(e3)

Доказ. Доказ леме 3.12 допуњен је додатним случајем индукције.

E је $\texttt{datatype(E}_1\texttt{)}$:

- $[[\texttt{E}_1]]_\mu = \texttt{err}$
  Из индуктивне хипотезе важи $[[\texttt{E}_1]]_\mu = \texttt{err}$ ако и само ако $[[\sigma(\texttt{E}_1)]]_\nu = \texttt{err}$.

$$
\begin{aligned}
[[\texttt{datatype(E}_1\texttt{)}]]_\mu &= \\
\text{(по дефиницији 3.8(e3))} &= \texttt{err} \\
\text{(по индуктивног хипотези)} &= [[\sigma(\texttt{E}_1)]]_\nu \\
\text{(по дефиницији 3.29(e3))} &= [[datatype(\sigma(\texttt{E}_1))]]_\nu \\
\text{(по дефиницији 3.20(e3))} &= [[\sigma(\texttt{datatype(E}_1\texttt{)})]]_\nu
\end{aligned}
$$

- $[[\texttt{E}_1]]_\mu \neq \texttt{err}$ и $[[\texttt{E}_1]]_\mu \notin \texttt{L}$:
  Из индуктивне хипотезе важи $[[\sigma(\texttt{E}_1)]]_\nu \notin \texttt{L}$.

$$
\begin{aligned}
[[\texttt{datatype(E}_1\texttt{)}]]_\mu &= \\
\text{(по дефиницији 3.8(e3))} &= \texttt{err} \\
\text{(по дефиницији 3.26(e3))} &= \texttt{DATATYPE}([[\sigma(\texttt{E}_1)]]_\nu) \\
\text{(по дефиницији 3.26(e3))} &= \mathcal{I}^{\mathcal{D}}(datatype)([[\sigma(\texttt{E}_1)]]_\nu) \\
\text{(по дефиницији 3.29(e3))} &= [[datatype(\sigma(\texttt{E}_1))]]_\nu \\
\text{(по дефиницији 3.20(e3))} &= [[\sigma(\texttt{datatype(E}_1\texttt{)})]]_\nu
\end{aligned}
$$

- $[[\texttt{E}_1]]_\mu \neq \texttt{err}$ и $[[\texttt{E}_1]]_\mu \in \texttt{L}$:
  Из индуктивне хипотезе важи $[[\texttt{E}_1]]_\mu \neq \texttt{err}$ ако и само ако $[[\sigma(\texttt{E}_1)]]_\nu \neq \texttt{err}$.

$$
\begin{aligned}
[[\texttt{datatype(E}_1\texttt{)}]]_\mu &= \\
\text{(по дефиницији 3.8(e3))} &= \texttt{dt}([[(\texttt{E}_1)]]_\mu) \\
\text{(по индуктивној хипотези)} &= \texttt{dt}([[\sigma(\texttt{E}_1)]]_\nu) \\
\text{(по дефиницији 3.26(e3))} &= \texttt{DATATYPE}([[\sigma(\texttt{E}_1)]]_\nu) \\
\text{(по дефиницији 3.26(e3))} &= \mathcal{I}^{\mathcal{D}}(datatype)([[\sigma(\texttt{E}_1)]]_\nu) \\
\text{(по дефиницији 3.29(e3))} &= [[datatype(\sigma(\texttt{E}_1))]]_\nu \\
\text{(по дефиницији 3.20(e3))} &= [[\sigma(\texttt{datatype(E}_1\texttt{)})]]_\nu
\end{aligned}
$$

□

Лема 3.14(e3)

Доказ. Доказ леме 3.14 допуњен је додатним случајем индукције.

R је `isliteral(E)`:

$\mu \Vdash$ `isliteral(E)`	акко
(по дефиницији 3.9(e3))	акко $[[\mathtt{E}]]_\mu \neq \mathtt{err}$ и $[[\mathtt{E}]]_\mu \in \mathtt{L}$
(по леми 3.12)	акко $[[\sigma(\mathtt{E})]]_\nu \neq \mathtt{err}$ и $[[\sigma(\mathtt{E})]]_\nu \in \mathtt{L}$
(по дефиницији 3.26(e3)	акко $\mathtt{ISLITERAL}([[\sigma(\mathtt{E})]]_\nu)$
(по дефиницији 3.26(e3)	акко $\mathcal{I}^{\mathcal{D}}(isliteral)([[\sigma(\mathtt{E})]]_\nu)$
(по дефиницији 3.27)	акко $(\mathfrak{D}, \nu) \models isliteral(\sigma(\mathtt{E}))$
(по дефиницији 3.20(e3))	акко $(\mathfrak{D}, \nu) \models \sigma(\mathtt{isliteral(E)})$

□

На начин сличан приказаном, могуће је даље проширити изразе и услове додавањем аритметичких или релационих оператора над изразима, уграђених функција за рад са бројевима, нискама, или функција са више од једног аргумента.

3.6.4 Аксиоме опште намене и моделовање RDF схеме

Дефиниција RDF скупа података не намеће никакво ограничење у односу између именованих и подразумеваног графа. Ако упит не прецизира упитни скуп података D својим `from` и `from named` клаузулама, за евалуацију упита користи се подразумевани граф, док његов садржај различите имплементације језика SPARQL третирају на различите начине. Врло често се сви триплети из свих именованих графова подразумевају да постоје такође и у подразумеваном графу. Аксиома опште намене *Default graph* додаје се са циљем моделовања ове зависности: ако је триплет `tp` присутан у неком именованом графу G, присутан је и у подразумеваном графу RDF скупа података $\mathbb{D}$.

RDF схема (енгл. RDF SCHEMA) намеће ограничења на интерпретацију графова и природно се моделује аксиомама. Зато, ако се садржаност упита анализира узимајући у обзир RDF схему, свака ставка схеме моделује се одговарајућом аксиомом, представљеном на слици 3.20. Ставке схеме могу бити следећег облика:

- `SubC rdfs:subClassOf C`
  Значење: Класа `SubC` је подкласа класе `C`, тј. свака инстанца класе `SubC` је такође и инстанца класе `C`. Ово својство моделује се аксиомом *Subclass*.

- `SubP rdfs:subPropertyOf P`
  Значење: Својство `SubP` је подсвојство својства `P`, тј. свака инстанца са својством `SubP` које је једнако некој вредности, има својство `P` исте вредности. Ово својство моделује се аксиомом *Subproperty*.

- `P rdfs:domain C`
  Значење: Домен својства `P` је скуп инстанци класе `C`, тј. ако нека инстанца има својство `P`, она мора бити инстанца класе `C`. Ово својство моделује се аксиомом *Domain*.

- `P rdfs:range C`
  Значење: Кодомен својства `P` је скуп инстанци класе `C`, тј. ако нека инстанца има својство `P`, вредност тог својства мора бити инстанца класе `C`. Ово својство моделује се аксиомом *Range*.

$$\forall\, s,p,o,i \quad \beta_n(s,p,o,i) \Rightarrow \beta_d(s,p,o) \qquad (\textit{Default graph})$$

$$\begin{aligned} &\forall\, s,i \quad \beta_n(s,\sigma(\texttt{a}),\sigma(\texttt{SubC}),i) \Rightarrow \beta_n(s,\sigma(\texttt{a}),\sigma(\texttt{C}),i) \\ &\forall\, s \quad\;\; \beta_d(s,\sigma(\texttt{a}),\sigma(\texttt{SubC})) \Rightarrow \beta_d(s,\sigma(\texttt{a}),\sigma(\texttt{C})) \end{aligned} \qquad (\textit{Subclass})$$

$$\begin{aligned} &\forall\, s,o,i \quad \beta_n(s,\sigma(\texttt{SubP}),o,i) \Rightarrow \beta_n(s,\sigma(\texttt{P}),o,i) \\ &\forall\, s,o \quad\;\; \beta_d(s,\sigma(\texttt{SubP}),o) \Rightarrow \beta_d(s,\sigma(\texttt{P}),o) \end{aligned} \qquad (\textit{Subproperty})$$

$$\begin{aligned} &\forall\, s,o,i \quad \beta_n(s,\sigma(\texttt{P}),o,i) \Rightarrow \beta_n(s,\sigma(\texttt{a}),\sigma(\texttt{C}),i) \\ &\forall\, s,o \quad\;\; \beta_d(s,\sigma(\texttt{P}),o) \Rightarrow \beta_d(s,\sigma(\texttt{a}),\sigma(\texttt{C})) \end{aligned} \qquad (\textit{Domain})$$

$$\begin{aligned} &\forall\, s,o,i \quad \beta_n(s,\sigma(\texttt{P}),o,i) \Rightarrow \beta_n(o,\sigma(\texttt{a}),\sigma(\texttt{C}),i) \\ &\forall\, s,o \quad\;\; \beta_d(s,\sigma(\texttt{P}),o) \Rightarrow \beta_d(o,\sigma(\texttt{a}),\sigma(\texttt{C})) \end{aligned} \qquad (\textit{Range})$$

Слика 3.20: Аксиоме за моделовање подразумеваног графа у скупу података и ставки RDF схеме

Како су `a`[7], `C`, `SubC`, `P`, и `SubP` IRI-јеви, тј. $\texttt{a},\texttt{C},\texttt{SubC},\texttt{P},\texttt{SubP} \in \texttt{I}$, функција σ је дефинисана над њима, тј. $\sigma(\texttt{a})$, $\sigma(\texttt{C})$, $\sigma(\texttt{SubC})$, $\sigma(\texttt{P})$ и $\sigma(\texttt{SubP})$ које се појављују у аксиомама су одговарајуће константе из скупа $\mathcal{C}$.

Иако сагласност и потпуност приступа нису доказани у случају аксиоми за моделовање RDF схеме, може се приметити да предложене аксиоме природно прате одговарајућу SPARQL семантику.

[7]Предикат у RDF схеми који се користи да искаже да је неки ресурс инстанца неке класе је `rdf:type`, а његова често коришћена скраћеница је `a`.

3.7 Релација стапања

Уместо садржаности упита, многи аутори разматрају релацију *стапања* (енгл. *subsumtion*) [9, 125, 149, 151], која се може тумачити као слабија форма релације садржаности, тј. њен надскуп, и која је у пракси врло прихватљива имајући у виду могуће непотпуне резултате SPARQL упита. Уместо како дефиниција 3.14 захтева да свако пресликавање μ из скупа $[\![Q_1]\!]^{\mathbb{D}}$ припада и скупу $[\![Q_2]\!]^{\mathbb{D}}$, релација стапања уводи се на наредном дефиницијом.

Дефиниција 3.14(c) (Стапање упита) Нека су Q_1 и Q_2 упити. Упит Q_1 *се стапа* у упит Q_2, у ознаци $Q_1 \dot{\sqsubseteq} Q_2$, ако за сваки RDF скуп података $\mathbb{D}$, за свако пресликавање μ из скупа $[\![Q_1]\!]^{\mathbb{D}}$ постоји његово проширење μ' ($\mu' \succeq \mu$), такво да μ' припада скупу $[\![Q_2]\!]^{\mathbb{D}}$.

Пример 3.27 Пример упита који се налазе у релацији стапања дат је на слици 3.21. Упити враћају студенте основних студија и њихова имена, при чему упит Q_2 враћа додатно и број телефона, док упит Q_1 само захтева његово постојање, али га игнорише.

Упит Q_1

```
select ?x ?n
where {
  ?x a       :UndergradStudent .
  ?x :name   ?n .
  ?x :phone ?tel
}
```

Упит Q_2

```
select ?x ?n ?tel
where {
  ?x a       :UndergradStudent .
  ?x :name   ?n .
  ?x :phone ?tel
}
```

Слика 3.21: Упит Q_1 (лево) се стапа у упит Q_2 (десно)

За разлику од релације садржаности, ако два упита задовољавају релацију стапања у оба смера, они не морају бити еквивалентни (ако било који од њих садржи `union` или оператор пројекције) [151].

При разматрању проблема стапања два упита, упит Q_2 може садржати пројекције, јер у овом случају, тај проблем није неодлучив као код проблема садржаности упита [151]. Такође, упити могу садржати све конструкте описане до сада. За оператор `union`, потребна је слабија форма леме 3.27, која разматра релацију $\dot{\sqsubseteq}$ уместо релације $\sqsubseteq$, чији је доказ аналоган.

Провера релације стапања своди се на исте провере на које се своди проблем садржаности упита, али где се релација $\sim$ (дефиниција 3.24) разматра у слабијој форми ($\subseteq$ уместо $=$), тј. проверава се релација $\dot{\sim}$ која се уводи наредном дефиницијом.

Дефиниција 3.24(c) (Релација $\dot{\sim}$) Упити Q_1 и Q_2 су у релацији $\dot{\sim}$, у ознаци $Q_1 \dot{\sim} Q_2$, ако је скуп релевантних променљивих $\overline{\texttt{rv}_1}$ упита Q_1 подскуп скупа релевантних променљивих $\overline{\texttt{rv}_2}$ упита Q_2, тј. ако важи

$$\overline{\texttt{rv}_1} \subseteq \overline{\texttt{rv}_2}.$$

Следећа лема је дуална леми 3.8.

Лема 3.8(c) *Нека су Q_1 и Q_2 упити такви да важи $Q_1 \dot{\sim} Q_2$. Нека је $\overline{rv_1}$ скуп релевантних променљивих упита Q_1, а $qpat_2$ и $\overline{dv_2}$ упит образац и скуп пројектованих променљивих упита Q_2, редом. Тада важи:*

$$\overline{rv_1} \subseteq var(qpat_2) \cap \overline{dv_2}.$$

Доказ. По дефиницији 3.24(c), из $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$, следи

$$\overline{\mathtt{rv}_1} \subseteq \overline{\mathtt{rv}_2}.$$

Тада, по леми 3.2, важи

$$\overline{\mathtt{rv}_1} \subseteq \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2}.$$

□

Теореме сагласности и потпуности (дуалне теоремама 3.18 и 3.22) могу се доказати и у овом случају.

Теорема 3.18(c) (Сагласност) *Ако је*

(1)	Θ *ваљана,*
или је	
(2)	$Q_1 \dot{\sim} Q_2$ *и* Ψ *је ваљана,*
тада важи	$Q_1 \dot{\sqsubseteq} Q_2$.

Доказ. Овај доказ је идентичан доказу теореме 3.18, од почетка до примене леме 3.8. Уместо ње, у овом случају, примењује се њена дуална лема 3.8(c), и из $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$ следи

$$\overline{\mathtt{rv}_1} \subseteq \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2},$$

тј.

$$\sigma(\overline{\mathtt{rv}_1}) \subseteq \sigma(\mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2}),$$

па по дефиницији 3.18 важи

$$\sigma(\overline{\mathtt{rv}_1}) \subseteq \sigma(\mathtt{var}(\mathtt{qpat}_2)) \cap \sigma(\overline{\mathtt{dv}_2}).$$

Тада, по леми 3.6, важи и

$$\overline{rv_1} \subseteq var(\sigma(\mathtt{qpat}_2)) \cap \overline{dv_2}, \tag{3.14}$$

Остатак доказа следи:

Нека ν' означава било које проширење валуације ν, такво да је

$$dom(\nu') = dom(\nu) \cup (var(\sigma(\mathtt{qpat}_2)) \setminus \overline{ov_2}).$$

По дефиницијама 3.22 и 3.25, како је формула Ψ ваљана, важи

$$(\mathfrak{D}, \nu') \models \forall \overline{rv_1}\ (\Phi_1(\overline{rv_1}) \Rightarrow \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)).$$

Како је ν' дефинисана на свим слободним променљивама из формуле

$$\Phi_1(\overline{rv_1}) \Rightarrow \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2),$$

тада важи:

$$(\mathfrak{D}, \nu') \models \Phi_1(\overline{rv_1}) \Rightarrow \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Из $(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv_1})$, како је $\nu' \succeq \nu$, следи да важи $(\mathfrak{D}, \nu') \models \Phi_1(\overline{rv_1})$. Зато, важи и

$$(\mathfrak{D}, \nu') \models \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Тада постоји проширење ν'' валуације ν' такво да је $dom(\nu'') = dom(\nu') \cup \overline{ov_2}$ и

$$(\mathfrak{D}, \nu'') \models \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Домен тог пресликавања је:

$$\begin{array}{ll} dom(\nu'') & = \\ \text{(по конструкцији)} & = dom(\nu) \cup (var(\sigma(\mathtt{qpat}_2)) \setminus \overline{ov_2}) \cup \overline{ov_2} \\ & = dom(\nu) \cup var(\sigma(\mathtt{qpat}_2)) \cup \overline{ov_2} \\ \text{(по дефиницији 3.22)} & = dom(\nu) \cup var(\sigma(\mathtt{qpat}_2)) \cup (var(\sigma(\mathtt{qpat}_2)) \setminus \overline{rv}_1) \\ & = dom(\nu) \cup var(\sigma(\mathtt{qpat}_2)) \\ \text{(јер је } dom(\nu) = \overline{rv_1}) & = \overline{rv_1} \cup var(\sigma(\mathtt{qpat}_2)) \\ \text{(због (3.14))} & = var(\sigma(\mathtt{qpat}_2)). \end{array}$$

Тада, по леми 3.15, за пресликавање μ'' такво да је $\mu'' \leftrightarrow \nu''$, важи

$$\mu'' \in [\![\mathtt{qpat}_2]\!]^{\mathtt{D}_2}_{df(\mathtt{D}_2)}.$$

По леми 3.1, $dom(\mu'') = \mathtt{var}(\mathtt{qpat}_2)$. По дефиницији 3.11, важи $\mu''_{\overline{\mathtt{dv}_2}} \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$, и

$$dom(\mu''_{\overline{\mathtt{dv}_2}}) = \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2}.$$

По дефиницијама 3.5 и 3.28, из $\mu \leftrightarrow \nu$, $\mu'' \leftrightarrow \nu''$ и $\nu \preceq \nu''$, следи $\mu \preceq \mu''$. Дакле и μ и $\mu''_{\overline{\mathtt{dv}_2}}$ су рестрикције пресликавања μ''. По дефиницији 3.24(c), како је $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$, важи $\overline{\mathtt{rv}_1} \subseteq \overline{\mathtt{rv}_2}$. Тада по леми 3.2 важи

$$\mathtt{var}(\mathtt{qpat}_1) \cap \overline{\mathtt{dv}_1} \subseteq \mathtt{var}(\mathtt{qpat}_2) \cap \overline{\mathtt{dv}_2},$$

тј. $dom(\mu) \subseteq dom(\mu''_{\overline{\mathtt{dv}_2}})$, па важи

$$\mu \preceq \mu''_{\overline{\mathtt{dv}_2}}.$$

Дакле, за сваки елемент скупа $[\![\mathtt{Q}_1]\!]^{\mathbb{D}}$ постоји проширење које припада скупу $[\![\mathtt{Q}_2]\!]^{\mathbb{D}}$ за било који скуп података $\mathbb{D}$. По дефиницији 3.14(c), важи

$$\mathtt{Q}_1 \dot{\sqsubseteq} \mathtt{Q}_2.$$

□

Теорема 3.22(c) (Потпуност) *Ако важи* $\mathtt{Q}_1 \dot{\sqsubseteq} \mathtt{Q}_2$, *тада*

(1) Θ *је ваљана,*

или

(2) $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$ *и* Ψ *је ваљана.*

Доказ. У случају када је $\mathtt{Q}_1$ незадовољив упит, доказ је идентичан као доказ теореме 3.22. У супротном, доказ $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$ ради се на сличан начин као у доказу теореме 3.22: Како је $\mathtt{Q}_1$ задовољив упит, по дефиницији 3.13, постоји скуп података $\mathbb{D}$ и пресликавање μ такво да

$$\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}.$$

По дефиницији 3.14(c), из $\mathtt{Q}_1 \dot{\sqsubseteq} \mathtt{Q}_2$ следи да постоји пресликавање μ', које је проширење пресликавања μ ($dom(\mu) \subseteq dom(\mu')$) такво да

$$\mu' \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}.$$

По дефиницији 3.12, важи $dom(\mu) = \overline{\mathtt{rv}_1}$ и $dom(\mu') = \overline{\mathtt{rv}_2}$. Дакле, важи

$$\overline{\mathtt{rv}_1} \subseteq \overline{\mathtt{rv}_2},$$

тј. по дефиницији 3.24(c), $\mathtt{Q}_1 \dot{\sim} \mathtt{Q}_2$.

У остатку доказа, све до примене леме 3.8, следеће промене су неопходне. Уместо задовољивости (3.6), (3.7), (3.8), (3.9) и (3.10), у случају стапања упита, како упит $\mathtt{Q}_2$ може садржати пројекције, важи редом:

$$\begin{aligned}
&\neg\Big(\forall \overline{rv_1}\big(\Phi_1(\overline{rv}_1) \Rightarrow \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\big)\Big)\\
&\exists \overline{rv_1}\ \big(\Phi_1(\overline{rv}_1) \wedge \neg(\exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2))\big)\\
\mathfrak{D} \models\ &\exists \overline{rv_1}\ \big(\Phi_1(\overline{rv}_1) \wedge \forall \overline{ov_2}\ \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\big)\\
(\mathfrak{D}, \nu) \models\ &\Phi_1(\overline{rv}_1) \wedge \forall \overline{ov_2}\ \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)\\
(\mathfrak{D}, \nu) \models \Phi_1(\overline{rv}_1) \quad &\text{и} \quad (\mathfrak{D}, \nu) \models \forall \overline{ov_2}\ \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)
\end{aligned}$$

Остатак доказа следи:

По дефиницији 3.14(c), из $\mu \in [\![\mathtt{Q}_1]\!]^{\mathbb{D}}$ и $\mathtt{Q}_1 \dot{\sqsubseteq} \mathtt{Q}_2$, постоји пресликавање μ' такво да је $\mu \preceq \mu'$ и $\mu' \in [\![\mathtt{Q}_2]\!]^{\mathbb{D}}$. По леми 3.16, за валуацију ν', такву да је $\mu' \leftrightarrow \nu'$, важи

$$(\mathfrak{D}, \nu') \models \Phi_2(\overline{rv_2}),$$

тј. по дефиницији 3.22,

$$(\mathfrak{D}, \nu') \models \exists \dot{\overline{ov_2}}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2),$$

где је $\dot{\overline{ov_2}} = var(\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2)) \setminus \overline{rv_2}$. Може се приметити да због дефиниција 3.5 и 3.28, како је $\mu \leftrightarrow \nu$, $\mu' \leftrightarrow \nu'$, и $\mu \preceq \mu'$, важи $\nu \preceq \nu'$. Тада постоји проширење ν'' валуације ν' на променљиве из $\dot{\overline{ov_2}}$, такво да важи

$$(\mathfrak{D}, \nu'') \models \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2),$$

али и

$$(\mathfrak{D}, \nu'') \models \exists \overline{ov_2}\ \sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Из $\nu \preceq \nu'$ и $\nu' \preceq \nu''$, ν је рестрикција валуације ν''. Зато важи

$$(\mathfrak{D}, \nu'') \models \forall \overline{ov_2}\ \neg\sigma^{df(\mathtt{D}_2)}(\mathtt{qpat}_2).$$

Ово је контрадикција, јер је $(\mathfrak{D}, \nu'')$ модел и формуле и њене негације, па је формула Ψ ваљана. □

3.8 Имплементација и експериментална евалуација

Предложени приступ је имплементиран у виду решавача SpeCS, оригиналног алата који је јавно доступан и отвореног је кода [178]. У овом поглављу, представљени су имплементациони детаљи и његова детаљна евалуација која садржи експерименталне резултате и поређење са релевантним савременим алатима.

3.8.1 Имплементациони детаљи

SpeCS је имплементиран у језику C++ (око 2,500 линија кода). За потребе лексичке и синтаксне анализе, коришћени су алати *Lex* и *Bison*. Подржани су упити описани упрошћеном граматиком датом на слици 3.1, њеним проширењима из поглавља 3.3, 3.4, 3.5 и 3.6. Такође, подржано је и неколико других синтаксних конструката, који су семантички еквивалентни претходним. На пример, могуће је груписање триплет образаца који деле исти субјекат, тако што се он наводи само једном иза чега следи листа његових предиката и одговарајућих објеката (енгл. *predicate-object list*), или образаца који деле исти и субјекат и предикат на сличан начин додавањем листе објеката (енгл. *object list*). Решавач је модуларан и подршка за додатно моделовање може се додати лако.

SpeCS генерише услове садржаности на основу улазних упита. Ако је RDF схема задата, генеришу се и аксиоме које одговарају конкретним правилима схеме. Све формуле генерисане су у SMT-lib формату [16], па се за њихово доказивање (оповргавање) могу користити различити решавачи/доказивачи. У експерименталној евалуацији, користио се SMT решавач *Z3* [67] (верзија 4.5.0) и FOL доказивач *Vampire* [158] (верзија 4.2.2). Решавач *Z3* показао је боље перформансе (око 50% је био бржи[8]), па су зато у наставку представљена времена решавања само уз коришћење *Z3* решавача. Мерење карактеристика алата SpeCS (који је интегрисан са решавачем *Z3*) на скуповима примера који се користе касније у евалуацији показује да је 3.83% укупног времена проведено у парсирању упита, 4.03% на конструкцији формула, док је најзахтевнији део решавање формуле решавачем *Z3*, тј. преосталих 92.14%.

3.8.2 Скупови примера за тестирање коришћени у евалуацији

Постоје два различита скупа примера за тестирање садржаности SPARQL упита, *Query Containment Benchmark* [42] и *SQCFramework* [161, 162]. Оба се користе у евалуацији доступних решавача.

Query Containment Benchmark (QC Bench) садржи фиксиран број ручно написаних проблема садржаности упита [42]. Неки од њих су означени као позитивни, тј. релација садржаности између два упита тог проблема је задовољена, док су други означени као негативни проблеми. Скуп примера је подељен на три дела растуће тежине, где сваки покрива све већи број језичких конструката:

- Конјунктивни упити без пројекција (енгл. ***C**onjunctive **Q**ueries with **No Proj**ection* — CQNoProj*)*: Ова група примера садржи 20 тест примера (`nop1-nop20`), 9 позитивних и 11 негативних, где се упит обрасци састоје од више триплет образаца повезаних оператором `.`, тј. упити су конјунктивни, и не садрже пројекције (`select *`).

[8] Ово не треба да буде тумачено као поређење решавача *Z3* и *Vampire*.

- Унија конјунктивних упита са пројекцијама (енгл. ***U**nion of **C**onjunctive **Q**ueries with **Proj**ection* — UCQProj): Ова група примера садржи 28 тест примера (`p1-p28`), 13 позитивних и 15 негативних, где упити могу имати `union` обрасце и пројекције у `select` клаузулама.

- Унија конјунктивних упита са RDF схемом (енгл. ***U**nion of **C**onjunctive **Q**ueries under **RDFS** reasoning* — UCQrdfs): На пољу SPARQL конструката, ова група је слична претходној, али она разматра RDF резоновање, тј. релација садржаности се анализира узимајући у обзир RDF схему. Садржи 28 тест примера (`rdfs1-rdfs28`), 12 позитивних и 16 негативних.

Овај скуп примера садржи и један тест пример са цикличним упитима, али он није укључен ни у једну од претходне три групе. Од свих решавача проблема садржаности SPARQL упита описаних у поглављу 2.3.4, који се користе у експерименталној евалуацији, AFMU, TS и JSAG не разматрају цикличне упите, за разлику од SA који их подржава [203], али је њихово решавање знатно спорије од уобичајеног. Када се ради о решавачу SpeCS, он успешно решава овај тест пример, и време решавања је слично осталим временима потребним за решавање осталих примера.

У току евалуације SpeCS решавача на овом скупу примера, примећена су два проблема, и то у примерима `p24` и `rdfs21`:

p24: У овом примеру разматра се релација садржаности између упита `Q20b` и `Q20a` (слика 3.22). По спецификацији, упит `Q20b` би требало да је садржан у упиту `Q20a`. Међутим, како пресликавање решења упита `Q20b` које одговара другом операнду `union` оператора садржи више променљивих у свом домену него одговарајуће пресликавање решења упита `Q20a`, релација садржаности није задовољена. На овом тест примеру решавачи AFMU и TS прекорачују временско ограничење за решавање постављено на 20000ms, SA не подржава пројектоване променљиве па и не разматра пример `p24`, док се JSAG слаже са решавачем SpeCS да је овај пример погрешно класификован у позитивне примере.

rdfs21: У овом примеру разматра се релација садржаности између упита `Q41e` и `Q41a` узимајући у обзир RDF схему `C3` (слика 3.23). Овај пример је означен као позитиван, тј. упит `Q41e` би требало да је садржан у упиту `Q41a`. Међутим, како је `prefix` клаузула упита `Q41e` различита од одговарајуће у упиту `Q41a`, релација садржаности није присутна. Од осталих решавача, SA и JSAG не узимају у разматрање RDF схему, па овај пример није подржан, док AFMU и TS игноришу `prefix` клаузуле и класификују пример у складу са погрешном спецификацијом.

Ова два проблема елиминисана су променом очекиваних одговора за тест примере `p24` и `rdfs21`, и додавањем још два позитивна тест примера, именована `p24a` и `rdfs21a`, заснована на примерима `p24` и `rdfs21`, али са пројекцијом `?x` у `select` клаузулама и истим префиксима у оба упита, редом.

Упит Q20b

```
prefix : <http://example.org/>
select * where {
   ?x a :Student .
   { ?x :name          ?y }
   union
   {
      ?x :nickName     ?z     .
      ?x :telephone    ?tel   .
      ?x :ssn          ?ssn   .
      ?x :sex          ?sex   .
      ?x :memberOf     ?dept  .
      ?x :emailAddress ?email .
      ?x :age          ?age   .
   }
   ?x :takesCourse ?course .
}
```

Упит Q20a

```
prefix : <http://example.org/>
select * where {
   ?x a :Student .
   { ?x :name          ?y     }
   union
   { ?x :nickName      ?z     }
   union
   { ?x :telephone     ?tel   }
   union
   { ?x :ssn           ?ssn   }
   union
   { ?x :sex           ?sex   }
   union
   { ?x :memberOf      ?dept  }
   union
   { ?x :emailAddress  ?email }
   union
   { ?x :age           ?age   }
   ?x :takesCourse ?course .
}
```

Слика 3.22: Тест пример p24 из *QC Bench*

SQCFramework је алат који генерише прилагођене скупове примера проблема садржаности SPARQL упита [161, 162] на основу реалних логова SPARQL упита. Може да генерише скупове различите величине који садрже SPARQL конструкте по жељи корисника[9], на пример, `union`, `optional`, `graph`, подупити, и друго. Такође, алат дозвољава кориснику да прецизира структурне карактеристике генерисаних SPARQL упита у смислу броја триплет образаца, броја пројекција, броја променљивих по којима се обрасци спајају, итд.

Аутори овог алата обезбедили су прегенерисане скупове примера за тестирање. Ови скупови су класификовани у две групе на основу логова упита који се користе за њихово генерисање:

- *DBpedia* и
- *Semantic Web Dog Food (SWDF).*

Свака група има 5 подгрупа, *KMeans++*, *DBSCAN+KMeans++*, *FEASIBLE*, *FEASIBLE-Exemplars* и *Random*, које се разликују по коришћеном алгоритму кластеровања. За сваку подгрупу, генерисани су скупови примера величине 15, 25, 50, 75, 100, и 125 (за *SWDF*), и 2, 4, 6, 9, 12, 15 (за *DBpedia* групу), где ове величине одговарају броју различитих над-упита у њима, док је укупан број тест примера у њима знатно већи.

[9]Свеобухватна анализа логова SPARQL упита са *DBPedia* триплет складишта показује да су неконјунктивни упити који садрже `union` и/или `optional` операторе врло коришћени у пракси [150], што оправдава ову могућност алата.

Упит Q41e

```
prefix : <http://www.example.org/>
select ?x where {
   { ?x :maleHeadOf   ?y . }
   union
   { ?x :femaleHeadOf ?y . }
}
```

Упит Q41a

```
prefix : <http:.../univ-bench.owl#>
select ?x where {
   ?x a :Professor .
}
```

Схема C3

```
prefix : <http://www.lehigh.edu//univ-bench.owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   :maleHeadOf    rdfs:subPropertyOf :headOf .
   :femaleHeadOf  rdfs:subPropertyOf :headOf .
   :FullProfessor rdfs:subClassOf    :Professor.
   :headOf        rdfs:domain        :FullProfessor .
```

Слика 3.23: Тест пример `rdfs21` из *QC Bench*

Укупно, цео корпус броји 78,359 тест примера (2,792 из *DBpedia* групе, и 75,567 из *SWDF* групе). Аутори тврде да су сви примери позитивни (релација садржаности упита је задовољена). Корпус је доступан на вебу у форми RDF података[10].

У једној трећини парова упита, релација садржаности јесте присутна, али у форми релације стапања. Такође, иако дефиниције из поглавља 3.1 претпостављају да су пројектоване променљиве из одговарајућих упита исто именоване, ово није случај у овом скупу примера за тестирање (један такав пример дат је на слици 3.24). Како је већина парова упита (93.63%) генерисана са различито именованим пројектованим променљивама, решавач SpeCS је унапређен и додата му је могућност резоновања и у оваквим ситуацијама. Са практичног аспекта, преименовање пројектованих променљивих је врло оправдано.

Евалуација решавача SpeCS потврђује да су 78.92% свих примера (61,840 тест примера) заиста позитивни тест примери. Међутим, за осталих 21.08% (16,519 тест примера, који су сви из *SWDF* групе), SpeCS закључује да релација садржаности упита није задовољена.[11] Ови примери су анализирани и разлози зашто релација садржаности није присутна класификовани су у следеће четири групе:

(1) У изразу у оквиру `filter` клаузуле, користи се променљива `?num` (уместо `?s`), иако она није додељена/коришћена у оквиру неког триплет обрасца (120 тест примера), на тај начин проузрокујући синтаксну грешку у упиту.

[10] Корпусу се може приступити преко адресе: `https://github.com/dice-group/sqcframework/blob/master/SQCFrameWork-benchmarks.7z`

[11] Аутори *SQCFramework* алата су обавештени и свесни ових проблема.

Упит Q_1

```
select ?type where {
   :Bracken_House rdf:type ?type
}
```

Упит Q_2

```
select ?concept where {
   ?s rdf:type ?concept
}
```

Слика 3.24: Упит Q_1 је под-упит упита Q_2, али пројектована променљива није исто именована.

(2) Над-упити имају конкретну вредност у `limit` клаузули, иако она није присутна, или има већу вредност у под-упиту (2,699 тест примера). Зато, у оваквим ситуацијама, релација садржаности не би требало да буде задовољена, јер нека решења над-упита могу бити искључена из његовог резултата, а у исто време присутна у резултату под-упита.

(3) Скуп графова одређен `from` клаузулама у под-упиту није подскуп скупа графова из над-упита (178 тест примера). У неким од ових примера, грешка у `prefix` клаузули је проузроковала разлике у `from` клаузулама упита, или се потпуно различити графови претражују, док у осталим примерима над-упит има једну `from` клаузулу, а под-упит не садржи ни једну приступајући подразумеваном графу.

(4) Под-упит није садржан у одговарајућем над-упиту (13,522 тест примера). Један такав пример, који демонстрира проблем који се јавља у многим примерима, представљен је на слици 3.25. Овде, под-упит користи граф образац у оквиру `optional` клаузуле, док је исти образац обавезан у над-упиту. Други пример, чија се форма такође среће у многим тест примерима, користи неки други IRI уместо предиката `rdf:type`.

Упит Q_1

```
select ?e ?t where {
   <http:.../2012> swc:hasLocation ?loc.
   ?e              swc:hasLocation ?loc
   filter ( ?e != <http:.../2012> )
   ?e              rdfs:label      ?lab
   optional { ?e   dc:title        ?t }
}
```

Упит Q_2

```
select ?s ?o where {
   ?s dc:title ?o
}
```

Слика 3.25: Упит Q_1 није под-упит упита Q_2.

За прву групу упита, синтаксне грешке исправљене су заменом променљиве `?num` променљивом `?s`. У вези са проблемима друге групе, иако SpeCS може открити такве ситуације, за потребе евалуације одлучено је да се SpeCS усагласи са осталим решавачима и игнорише `limit` клаузуле. Остали парови упита који не задовољавају релацију садржаности (13,700 тест примера), означени су као негативни примери, тј. промењена је спецификација скупова тест примера. Остали решавачи коришћени у евалуацији не подржавају већину ових негативних тест примера, док SA не подржава ни један (јер сви ови упити садрже пројектоване променљиве). AFMU и TS подржавају само 52 тест примера из групе (3), али их класификују као позитивне. JSAG подржава само 57 тест примера, од тога 25 из групе (3) и, слично као AFMU и TS, такође их класификује као позитивне, и 32 тест примера из групе (4), за које је тачно потврдио да релација садржаности упита није задовољена. AFMU, TS и JSAG нетачно решавају подржане тест примере из групе (3) због игнорисања прецизираних графова у `from` клаузулама упита.

Решавач SpeCS такође је покренут на свим овим тест примерима али у супротном смеру, са циљем истраживања колико над-упита је садржано у одговарајућим подупитима. Број таквих парова је 4,773, тј. 6.09% укупног броја.

3.8.3 Поређење са релевантним савременим решавачима

Како је описано у поглављу 2.3.4, постоји неколико савремених решавача проблема садржаности упита. Поређење свих релевантних карактеристика које решавачи подржавају или не, сумаризовано је у табели 3.2, док су детаљније информације о бројевима подржаних и успешно решених примера из представљених скупова за тестирање у експерименталној евалуацији дати табелом 3.3.

Табела 3.2: Поређење својстава подржаних савременим решавачима.

		SA	AFMU	TS	JSAG	SpeCS
SPARQL конструкт	конјунктивни упити	✓	✓	✓	✓	✓
	`union`		✓	✓	✓	✓
	неименовани чворови		✓	✓	✓	✓
	пројектоване променљиве		✓	✓	✓	✓
	`filter`		✓	✓	✓	✓
	`optional`	✓				✓
	подупити	✓	✓	✓		✓
	`graph`		✓	✓	✓	✓
	изрази у `select` клаузули					✓
	уграђене функције		✓	✓	✓	✓
	циклични упити	✓				✓
QC тип	релација стапања	✓			✓	✓
	RDF схема		✓	✓		✓
	преименовање пројекција				✓	✓
	сагласност & потпуност		✓	✓		✓

SA [125]: Решавач покрива добро засноване (енгл. *well-designed*) подкласе SPARQL граф образаца, који садрже, за разлику од приступа представљеног овде, мањи скуп језичких конструката (види табелу 3.2 за детаље). Анализирање релације садржаности упита узимајући у обзир RDF схему није подржано, за разлику од SpeCS решавача. SA стриктно поштује дефиницију релације стапања, па преименовање променљивих није дозвољено. Рад његових аутора је пре свега фокусиран на теоријско испитивање комплексности релација стапања и еквиваленције, док сагласност и потпуност приступа нису покривени.

AFMU [42]: Решавач своди проблем садржаности упита на проблем испитивања задовољивости, слично као у овде представљеном приступу. Међутим, он користи μ-рачун, чији су решавачи спорији од SMT решавача. Скуп покривених језичких конструката у AFMU решавачу је такође мањи у поређењу са решавачем SpeCS (види табелу 3.2 за детаље). AFMU не подржава релацију стапања и преименовање пројектованих променљивих, али је резоновање о релацији садржаности у односу на RDF схему могуће. Сагласност и потпуност приступа су доказани.

TS [92]: Приступ је сличан приступу примењеном у оквиру AFMU решавача јер и он користи μ-рачун, али са неким разликама у моделовању упита. Његове мане у односу на SpeCS приступ исте су као код претходног решавача.

JSAG [180]: Решавач не подржава неке конструкте доступне у SpeCS алату, нити садржаност у односу на RDF схему, али је резоновање о релацији стапања и преименовање променљивих могуће. За разлику од осталих приступа који су ограничени на проверу релације садржаности између два упита, JSAG је способан да генерише упит садржан у свим другим упитима из неког скупа. Сагласност и потпуност приступа нису доказани.

3.8.4 Експериментално окружење

Сви експерименти су извршавани на машини са *Intel® Core™ i7-4790 CPU @ 3.60GHz* процесором (8 језгара) и 16GB RAM меморије, са инсталираним *Ubuntu 18.04.3 LTS* оперативним системом.

Сва мерења могу се поновити у потпуности. Решавачи SA, AFMU, TS преузети су са веб странице која садржи скуп примера за тестирање *QC Bench* [41]. Са циљем евалуације решавача JSAG (који је доступан на *Github* сервису [179]) и решавача SpeCS [178], имплементирани су додатни *Java* омотачи. Могућност извршавања осталих скупова тест примера генерисаних *SQCFramework* алатом је такође додата. Сва ова унапређења јавно су доступна са детаљним резултатима евалуације (укључујући логове извршавања и измерена времена) и налазе се на вебу [178].

3.8.5 Резултати евалуације на *QC Bench* тест примерима

На слици 3.26 представљена су времена извршавања потребна решавачима за сваки појединачни тест пример успешно решен у задатом временском ограничењу од 20000ms, као и њихова просечна времена, по групама *CQNoProj*, *UCQProj* и *UCQrdfs*, редом.

Слика 3.26: Појединачна и просечна времена успешних решавања тест примера из група *CQNoProj*, *UCQProj* и *UCQrdfs*.

Горњи део табеле 3.3 садржи сумаран приказ евалуације над скуповима примера за тестирање *QC Bench*. Решавач SA најбржи је на примерима из групе *CQNoProj*, али је један тест пример решио погрешно, док додатни SPARQL конструкти присутни у другим групама примера нису подржани. Решавач JSAG не подржава резоновање о релацији садржаности са присутном RDF схемом. SpeCS је једини решавач који је решио све примере тачно (у задатом временском ограничењу и без нетачних резултата). Такође, ефикаснији је од решавача AFMU, TS и JSAG од 2 до 42 пута.

Табела 3.3: QC Bench и SQCFramework резултати. За сваку групу тест примера и за сваки решавач, представљени су бројеви који одговарају стопираним (временско ограничење је 20000ms), неподржаним, нетачно и тачно решеним, као и просечно време успешно решених тест примера у милисекундама. Најбоље вредности за сваки ред (укључујући ефикасност и тачност) су наглашене.

Група				AFMU	SA	TS	JSAG	SpeCS
QC Bench	CQNoProj		#TO/#Неподрж./#Нетач. #Тачни	4 / 0 / 0 16	0 / 0 / 1 19	**0 / 0 / 0** **20**	**0 / 0 / 0** **20**	**0 / 0 / 0** **20**
			Просек у ms	114.88	**5.12**	338.68	36.95	12.88
	UCQProj		#TO/#Неподрж./#Нетач. #Тачни	11 / 0 / 0 18	0 / 29 / 0 0	2 / 0 / 0 27	0 / 0 / 3 26	**0 / 0 / 0** **29**
			Просек у ms	63.15	-	883.87	45.22	**21.23**
	UCQrdfs		#TO/#Неподрж./#Нетач. #Тачни	0 / 0 / 2 27	0 / 29 / 0 0	0 / 0 / 1 28	0 / 29 / 0 0	**0 / 0 / 0** **29**
			Просек у ms	139.07	-	153.94	-	**19.59**
SQCFramework	DBPedia-Sup15	DBSCAN+ KMeans++	#TO/#Неподрж./#Нетач. #Тачни	0 / 130 / 0 16	0 / 146 / 0 0	0 / 130 / 0 16	0 / 127 / 0 19	**0 / 0 / 0** **146**
			Просек у ms	20.10	-	22.52	27.21	**16.48**
		FEASIBLE	#TO/#Неподрж./#Нетач. #Тачни	0 / 188 / 0 9	0 / 197 / 0 0	0 / 188 / 0 9	0 / 185 / 0 12	**0 / 0 / 0** **197**
			Просек у ms	22.43	-	22.23	24.22	**15.13**
		FEASIBLE-Examplars	#TO/#Неподрж./#Нетач. #Тачни	0 / 186 / 0 9	0 / 195 / 0 0	0 / 186 / 0 9	0 / 183 / 0 12	**0 / 0 / 0** **195**
			Просек у ms	21.85	-	23.11	26.66	**15.59**
		KMeans++	#TO/#Неподрж./#Нетач. #Тачни	0 / 130 / 0 16	0 / 146 / 0 0	0 / 130 / 0 16	0 / 127 / 0 19	**0 / 0 / 0** **146**
			Просек у ms	21.33	-	22.17	25.40	**15.09**
		Random	#TO/#Неподрж./#Нетач. #Тачни	0 / 176 / 0 4	0 / 180 / 0 0	0 / 176 / 0 4	0 / 176 / 0 4	**0 / 0 / 0** **180**
			Просек у ms	25.30	-	22.14	24.93	**15.30**
	SWDF-Sup125	DBSCAN+ KMeans++	#TO/#Неподрж./#Нетач. #Тачни	3 / 2794 / 3 227	0 / 3027 / 0 0	0 / 2764 / 3 260	0 / 2748 / 24 255	**0 / 0 / 0** **3027**
			Просек у ms	21.88	-	68.44	22.60	**15.96**
		FEASIBLE	#TO/#Неподрж./#Нетач. #Тачни	3 / 4889 / 5 252	0 / 5149 / 0 0	0 / 4884 / 5 260	0 / 4528 / 3 618	**0 / 0 / 0** **5149**
			Просек у ms	20.27	-	67.58	18.56	**16.22**
		FEASIBLE-Examplars	#TO/#Неподрж./#Нетач. #Тачни	3 / 4864 / 5 261	0 / 5133 / 0 0	0 / 4857 / 5 271	0 / 4528 / 3 602	**0 / 0 / 0** **5133**
			Просек у ms	20.62	-	65.98	19.00	**16.07**
		KMeans++	#TO/#Неподрж./#Нетач. #Тачни	3 / 3659 / 5 230	0 / 3897 / 0 0	0 / 3627 / 5 265	0 / 3176 / 23 698	**0 / 0 / 0** **3897**
			Просек у ms	22.25	-	67.63	19.49	**15.89**
		Random	#TO/#Неподрж./#Нетач. #Тачни	0 / 1466 / 2 54	0 / 1522 / 0 0	0 / 1461 / 2 59	0 / 1385 / 7 130	**0 / 0 / 0** **1522**
			Просек у ms	20.49	-	22.40	21.09	**14.50**

3.8.6 Резултати евалуације на *SQCFramework* тест примерима

Доњи део табеле 3.3 садржи сумарни приказ евалуације на скуповима тест примера генерисаних алатом *SQCFramework*. У сврху поређења, коришћена је само највећа подгрупа из сваке од 10 група представљених у поглављу 3.8.2, које укупно садрже 19,592 парова упита. Алат SA не подржава ни један од ових тестова, док остали решавачи, AFMU, TS и JSAG не подржавају већину њих. Укупни проценти успешно решених тест примера у задатом временском ограничењу од 20000ms опадајућим редоследом су:

(i) SpeCS - 100%[12],

(ii) JSAG - 12.09%,

(iii) TS - 5.97%,

(iv) AFMU - 5.50%,

(v) SA - 0.00%.

Осим што је решио све примере успешно (ред величине више од осталих), SpeCS је и најбржи решавач по свим групама тест примера (за фактор између 1.14 и 4.29).

[12]При имплементацији решавача SpeCS, циљ је био подржати све језичке конструкте присутне у овом скупу тест примера, што је резултирало процентном од 100% успешности. Како ни SpeCS не подржава све конструкте језика SPARQL (рецимо `bind` клаузулу) могуће је направити нов скуп примера за тестирање у коме ће такви конструкти бити заступљени и где SpeCS неће решити све примере.

Глава 4

Верификација у служби рефакторисања C/C++ кода са уграђеним SQL упитима

У претходној глави разматрана је релација садржаности, као најбитнија релација између два упита. Аутоматска провера њеног постојања омогућена је моделовањем упита и саме релације у теоријама логике првог реда. Предложено моделовање је у складу са одговарајућом семантиком упитног језика. Једна од примена таквог моделовања може бити при разматрању функционалне еквивалентности императивног кода који у себи има приступ бази података. Резоновање у овом контексту је далеко сложеније, јер укључује семантике и упитног језика и језика из кога се упити позивају, тј. језика који припадају различитим програмским парадигмама, декларативној и императивној.

Апликације за приступ базама података имају специфичне могућности за рефакторисање, јер обично укључују бар два различита програмска језика: упитни језик (најчешће SQL) и неки програмски језик опште намене, тј. матични језик. Унапређење кода у апликацијама за приступ базама података може подразумевати рефакторисање SQL упита, рефакторисање наредби матичног језика, и рефакторисање интеракције између њих, тј. симултане измене у оба дела кода које чувају глобалну еквивалентност (без чувања еквиваленције ових двају делова посматраних одвојено). Док постоје бројна истраживања на тему провере еквивалентности два SQL упита [51], и провере еквивалентности императивних програма [95], провера еквиваленције која покрива интеракцију између SQL упита и матичног језика скоро је непокривена у литератури, иако нуди значајне могућности. На пример, измештања рачунања из SQL-а у матични језик, и обратно, може омогућити употребу прекомпилираних упита, који су важни из угла ефикасности [73]. Главна препрека у оваквом истраживању је отежано дефинисање семантике која може да моделује императивну и декларативну природу ових двају језика и о којој се може ефикасно резоновати.

У овој глави, представљена је семантика SQL упита у логици првог реда заснована на аксиомама, која се може повезати са семантиком императивних језика и користити се за аутоматско резоновање о еквивалентности функција које садрже SQL упите [171, 173]. Новопредложени приступ, пре свега има за циљ доказивање еквивалентности у случајевима када ни SQL кôд, ни кôд на језику опште намене нису еквивалентни појединачно, али њихова комбинација јесте (видети слику 4.1). Дакле, овај приступ се може користити као додатак приступима који разматрају само императивну или само SQL еквивалентност.

Предложено решење имплементирано је као алат SQLAV за SQL упите који су уграђени у програмски језик C/C++, јавно је доступно и отвореног је кода [196]. Ова комбинација је изабрана јер има важних предности, које се углавном односе на проверу SQL синтаксе, типова и схеми у фази компилације, што нуди добре перформансе (SQL наредбе не морају бити провераване и поново компилиране у фази извршавања), тј. уграђивање SQL упита у C/C++ је добар избор за остваривање процедуралних могућности у апликацијама за приступ базама података [63]. Такође, уграђени SQL врло је популаран у уређајима са уграђеним рачунаром, јер нуди добру ефикасност, како у виду меморије, тако и потрошње енергије [108], што је јако битно у контексту ограничених могућности ових уређаја [73, 204]. Предложени приступ демонстрира могућности доказивања еквивалентности функција написаних у два различита језика, од којих један припада императивној, а други декларативној парадигми.

Преглед главе: Поглавље 4.1 представља један предвиђени случај коришћења предложеног алата. Поглавље 4.2 описује оригинално моделовање програма са уграђеним SQL кодом, док се у поглављу 4.3 представља предложени оквир (енгл. *framework*), тј. конструкција услова еквивалентности. У поглављу 4.4, дате су информације о имплементацији и евалуацији предложеног приступа.

4.1 Предвиђени случај употребе и други сценарији коришћења

У овом делу приказан је један сценарио коришћења предложеног приступа, који користи алат SQLAV за проверу исправности рефакторисања приступа бази која је условљена проблемом са перформансама. Одговарајући кôд је приказан на слици 4.1 и биће коришћен као текући пример до краја ове главе.

Пример 4.1 (Текући пример) Претпоставимо да су корисници банкарске апликације за прављење и анализу извештаја пријавили проблем са ефикасношћу при генерисању одређених докумената. Искусни програмер, коме је додељен овај задатак, идентификовао је проблем и закључио да већина времена одлази на приступ бази података, тј. у функцији која садржи позив SQL упита приказаној на слици 4.1 (лево).

Анализирајући упит, приметио је више рачунања у самом SQL упиту, што може негативно утицати на перформансе. На пример, израз `NOT (a_avail_blnc - :in2_sqc >= 0)` у `WHERE` делу упита може проузроковати спорије извршавање упита, јер се овај израз мора евалуирати за сваки ред из `account` табеле. Са друге стране, еквивалентни услов `a_avail_blnc <:in2_sqc` потенционално може користити индекс у бази података одржаван над атрибутом `a_avail_blnc` табеле `account`, осигуравајући да се евалуација израза не извршава за сваки ред табеле, већ се филтрирање редова врши приступом индексу, што је много ефикасније. Из сличних разлога, услов `t_txn_date >:in1_sqc - 14` би могао бити замењен условом `t_txn_date >:in1_sqc`, где је улазни параметар `:in1_sqc` смањен за 14 у оквиру C/C++ језика пре декларисања курсора. Програмер такође проналази да `ORDER BY` услови могу бити упрошћени, и тако смањити време потребно за сортирање резултата коришћењем погодног редоследа редова у одговарајућем индексу.

Добра програмерска пракса налаже да се само мале промене могу примењивати у оквиру корака рефакторисања, и да сваки сваки такав корак треба бити праћен темељ-

```
1  ...
2  in1_sqc = today;
3  in2_sqc = threshold;
4  EXEC SQL DECLARE c1 CURSOR FOR
5   SELECT a_avail_blnc + t_amount,
6     :in1_sqc - t_txn_date, t_txn_id,
7     a_account_id, a_cust_id,
8     c_cust_type
9   FROM account,transaction,customer
10  WHERE t_txn_date > :in1_sqc - 14
11    AND NOT
12     (a_avail_blnc-:in2_sqc >= 0)
13    AND a_cust_id = c_cust_id
14    AND a_account_id = t_account_id
15  ORDER BY -t_amount,
16    a_avail_blnc + t_amount,
17    :in1_sqc - t_txn_date, a_cust_id,
18    a_account_id, t_txn_id
19  LIMIT 5;
20 ...
21 int i = 0;
22 do {
23   EXEC SQL FETCH c1 INTO
24      :old_blnc_sqc, :days_sqc,
25      :txn_id_sqc, :account_id_sqc,
26      :cust_id_sqc, :cust_type_sqc;
27
28   ...
29   days[i] = days_sqc;
30   blnc[i] = old_blnc_sqc;
31
32   i++;
33 } while (i < 5);
```

```
1  ...
2  in1_sqc = today - 14;
3  in2_sqc = threshold;
4  EXEC SQL DECLARE c1 CURSOR FOR
5   SELECT c_cust_id, c_cust_type,
6     a_account_id, t_txn_id,
7     t_txn_date,
8     a_avail_blnc, t_amount
9   FROM customer,account,transaction
10  WHERE t_txn_date > :in1_sqc
11    AND a_avail_blnc < :in2_sqc
12
13    AND a_cust_id = c_cust_id
14    AND a_account_id = t_account_id
15  ORDER BY t_amount DESC,
16    a_avail_blnc,
17    t_txn_date DESC, c_cust_id,
18    a_account_id, t_txn_id
19  LIMIT 5;
20 ...
21 int i = 0;
22 do {
23   EXEC SQL FETCH c1 INTO
24      :cust_id_sqc, :cust_type_sqc,
25      :account_id_sqc, :txn_id_sqc,
26      :date_sqc, :avail_blnc_sqc,
27      :amount_sqc;
28   ...
29   days[i] = today - date_sqc;
30   blnc[i] = avail_blnc_sqc
31                + amount_sqc;
32   i++;
33 } while (i < 5);
```

Слика 4.1: Текући пример: Нађи пет трансакција са највећим износом реализованих у последње две недеље где је корисник потрошио скоро сва расположива средства са свог рачуна (вредност средстава на рачуну је мања од унапред задате минималне вредности). Рачунања су премештена из SQL кода (лево) у C/C++кôд (десно).

ним тестирањем [84]. Међутим, постоји много случајева који би требало да се провере за овај део кода и комплетно тестирање ових промена би одузело више времена од оног што програмер има на располагању. Зато он одлучује да користи систем SQLAV, као брз и поуздан алат за безбедно рефакторисање.

Програмер прво рефакторише SQL упит и одговарајућу фукцију, мењајући `WHERE` клаузулу и смањујући улазни параметар `in1_sqc` у C/C++ језику, тј. мења линије 2, 10, 11 и 12 на начин приказан на слици 4.1 (десно). Покретањем SQLAV алата, проверио је еквивалентност ове две верзије кода. Након тога, рефакторише `ORDER BY` клаузулу, како је показано на слици 4.1 (десно) у линијама 15, 16 и 17, али у линији 17 програмер прави грешку, тј. заборавља да стави кључну реч `DESC` којом захтева опадајући

поредак редова по вредностима атрибута `t_txn_date`. Поновним позивом SQLAV система, на основу исписане поруке упозорења открива грешку врло лако. Поправивши ово, програмер такође незнатно мења и `SELECT` део упита, како би потпуно елиминисао рачунске операције из SQL упита, променио редослед пројекција и табела у `FROM` клаузули. Мењајући линије 5–8, и 24–31, добија се кôд приказан на слици 4.1 (десно). SQLAV потврђује да је нови кôд еквивалентан полазном, па програмер зна да се може поуздати у ове разултате и да је одговарајуће тестирање у овом случају непотребно.

Сада, програмер треба да понови мерења и да потврди да унете промене доносе жељено убрзање, и у зависности од постигнутих резултата настави даље. Ако је одговор позитиван, програмер успешно завршава свој задатак, и алат SQLAV му је знатно уштедио време. Укупно време потребно алату SQLAV за верификацију еквиваленције у овом случају је 3.082 секунди. У приказаном сценарију, програмер је покретао SQLAV три пута, па је укупно време од десетак секунди свакако краће од извршавања тестова, где код сваке озбиљније апликације чак и један тест може да се извршава знатно дуже.

Представљени процес са малим изменама праћеним SQLAV проверама може се реализовати као један велики монолитни корак који укључује све поменуте измене, јер SQLAV даје прецизне информације у случају да се еквивалентност не може доказати (за више детаља, видети поглавље 4.3), па се грешке могу открити на једноставнији начин него приликом тестирања. Дакле, програмер може направити све жељене измене па покренути алат само једном, а SQLAV ће усмерити његову пажњу на проблематичне делове, ако они постоје.

Осим описаног сценарија, SQLAV може се применити и у разним другим ситуацијама у којима је битно одржати функционалну еквивалентност између две верзије кода, када су промене присутне у оба језика, и у C/C++ и у SQL упиту. Најбољи резултати могу се очекивати када се алат користи по једном за сваку промену у фукцији која садржи SQL упит, у духу генерално прихваћених препоруки рефакторисања кода. За такве промене, време потребно алату је углавном занемарљиво (за више детаља, видети поглавље 4.4), па се SQLAV може интегрисати у развојно окружење, и тако бити лако доступан у развојном процесу. Алат SQLAV може се користити на различите начине: као самосталан алат, или у оквиру окружења, тј. као део *GitHub Actions* процеса, где се може конфигурисати да се извршава на експлицитан захтев програмера, или аутоматски после сваког уношења измена у репозиторијум (за више детаља, видети поглавље 4.4.1).

Две важне предности коришћења алата SQLAV у овом контексту су:

- добијање прецизних и информативних порука у случају да се еквивалентност не може доказати, и
- већа поузданост резултата, у супротном случају.

Главни недостаци алата SQLAV су:

- неподржаност комплетног језика SQL,
- ограничен скуп подржаних типова података,
- сагласност и потпуност приступа нису доказани.

И поред поменутих недостатака, SQLAV може бити користан, како је показано у овом поглављу, и може се користити као допуна процесу тестирања у реалним ситуацијама, доприносећи већој поузданости у односу само на тестирање и лакшем откривању узрока одсуства еквиваленције у рефакторисаном коду.

4.2 Моделовање уграђених SQL упита

У овом поглављу, описан је приступ за моделовање уграђених SQL упита, илустрован различитим примерима. Такође, дискутују се донете одлуке узроковане ефикасношћу аутоматског резоновања о еквивалентности кода.

У наставку, разматрани су SQL упити чија је структура статички дефинисана. Уграђени упити могу се користити у два контекста:

(1) Функција у коју се уграђује упит очекује од њега једну повратну енторку. У овом случају користе се `select ... into:` конструкти. Овакви упити ће се у даљем тексту називати *s-упити* (*s-* од енгл. *single row*). Ако се очекује енторка дужине 1, тј. једна вредност, упит ће бити означен као *su-упит* (*u-* од енгл. *unit*), док ће се у супротном користити назив *st-упит* (*t-* од енгл. *tuple*).

Пример 4.2 На слици 4.2 представљена су два упита са `select ... into:` конструктима, па су оба *s*-упити. Ипак, упит са леве стране је *su*-упит, јер издваја само једну вредност у променљиву `:pending_deposit_sqc`, док је упит са десне стране *st*-упит јер очекује енторку дужине 2, (тј. две вредности) и смешта их у променљиве `:pending_blnc_sqc` и `:avail_blnc_sqc`.

```
1  int pending_deposit(int account_id){
2   account_sqc = account_id;
3
4   EXEC SQL SELECT T1.a_pending_blnc
5                   -T1.a_avail_blnc
6    INTO :pending_deposit_sqc
7
8    FROM account AS T1
9    WHERE T1.a_account_id=:account_sqc;
10
11  if (sqlca.sqlcode) goto errorexit;
12
13  return pending_deposit_sqc;
14
15 }
```

```
1  int pending_deposit(int account_id){
2   account_sqc = account_id;
3
4   EXEC SQL SELECT T1.a_pending_blnc,
5                   T1.a_avail_blnc
6    INTO :pending_blnc_sqc,
7         :avail_blnc_sqc
8    FROM account AS T1
9    WHERE T1.a_account_id=:account_sqc;
10
11  if (sqlca.sqlcode) goto errorexit;
12
13  return pending_blnc_sqc
14         -avail_blnc_sqc;
15 }
```

Слика 4.2: Рачунање над повратим вредностима је урађено у SQL упиту (лево), и премештено је у C/C++ (десно).

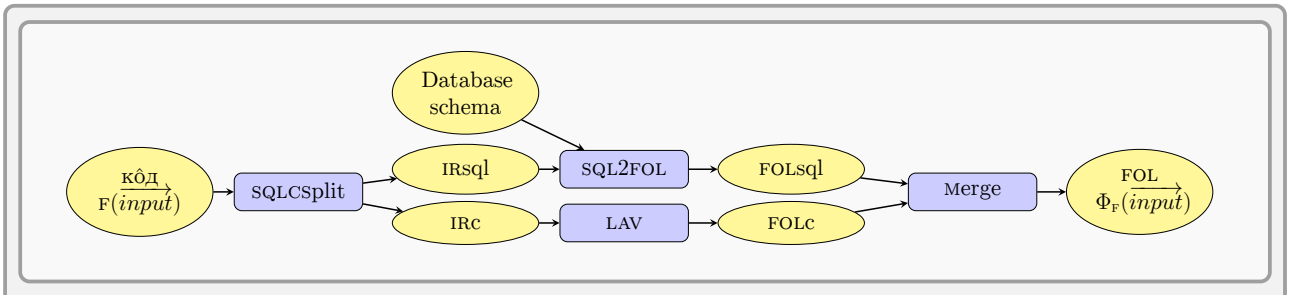
(2) Функција у коју се уграђује упит очекује од њега више повратних енторки. У овом случају, користе се курсори. Овакви упити називаће се *m-упити* (*m-* од енгл. *multiple*), или прецизније *mu-упити* ако су енторке дужине 1, тј. *mt-упити* за енторке веће дужине.

Пример 4.3 На слици 4.1 представљена су два упита са курсорима, па су оба *m*-упити. Како се у оба издвајају енторке дужине веће од 1 (у упиту на левој страни енторке су дужине 6, а на десној дужине 7), ови упити су *mt*-упити.

Пратећи семантику уграђеног SQL-а, упит може да се или изврши успешно (и врати неке податке), или да из неког разлога нема података у бази који задовољавају тражене услове. Да би s-упит био успешно извршен, мора постојати јединствен ред у бази који задовољава све услове. У случају да се m-упит извршава, мора постојати бар један такав ред, тј. не постоји ограничење јединствености. Због оваквих разлика, s-упити и m-упити морају се моделовати на различите начине.

Анализа кода пролази кроз четири фазе (слика 4.3):

1. SQLCSplit: подела оригиналног кода $\text{F}(\overrightarrow{input})$ у међурепрезентације IRsql и IRc, што ће бити објашњено у поглављу 4.2.1,
2. LAV: трансформација међурепрезентације IRc у FOLc формулу, што ће бити објашњено у поглављима 4.2.2,
3. SQL2FOL: трансформација међурепрезентације IRsql на основу схеме базе података у скуп аксиома FOLsql, што ће бити објашњено у поглављу 4.2.3, и
4. Merge: спајање FOLc формуле и скупа аксиома FOLsql у јединствену FOL формулу $\Phi_{\text{F}}(\overrightarrow{input})$, што ће бити објашњено у поглављу 4.2.4.



Слика 4.3: Трансформација изворног кода у FOL формулу кроз четири фазе: SQLCSplit, LAV, SQL2FOL и Merge.

4.2.1 Подела оригиналног кода у међурепрезентације

Уграђени SQL може садржати различите директиве, макрое и функције. На пример, директиве за укључивање могућности обраде SQL грешака и макрои за штампање одговарајућих порука у фази SQLCSplit се елиминишу. Главни циљ је доказивање еквиваленције две функције са уграђеним SQL упитима, па је претпоставка да ови други делови имају исту очекивану функционалност оправдана.

Сваки уграђени SQL упит директно се издваја у своју међурепрезентацију IRsql, која се разликује од уобичајеног SQL-а по томе што садржи променљиве из C/C++ језика.

У случају када се користе погледи (енгл. *table views*) у оквиру SQL кода, неопходна је додатна фаза у трансформацији SQL-а, пре него што се настави са следећом. Ова фаза ће бити реферисана као SQL *претпроцесирање*. У оквиру ње, кôд се трансформише у еквивалентан који их не садржи, тј. упити у којима се користе погледи се замењују одговарајућим без њих. Овај приступ подржава елиминацију простих погледа, тј. погледа који не садрже `ORDER BY` ни `LIMIT` клаузуле.

Пример 4.4 Слика 4.4 садржи пример SQL претпроцесирања. Табеле које се спајају у погледу додате су у FROM клаузуле упита који користи поглед, док су услови из WHERE клаузуле погледа додати условима филтрирања у самом упиту.

```
EXEC SQL CREATE VIEW V1 AS
 SELECT T1.a_account_id,
        T1.a_avail_blnc,
        T2.c_cust_id, T3.i_fname,
        T3.i_lname
 FROM account AS T1,
      customer AS T2,
      individual AS T3
 WHERE T1.a_cust_id = T2.c_cust_id
   AND T2.c_cust_id = T3.i_cust_id;
...
```

```
...
EXEC SQL SELECT P.i_lname
         INTO :output_sqc
  FROM V1 AS P,
       transaction AS T1
  WHERE T1.t_txn_id = :input_sqc
    AND T1.t_account_id =
        P.a_account_id;
...
```

```
EXEC SQL SELECT T3.i_lname INTO :output_sqc
 FROM account AS T1, customer AS T2, individual AS T3, transaction AS T4
 WHERE T1.a_cust_id = T2.c_cust_id AND T2.c_cust_id = T3.i_cust_id
   AND T4.t_txn_id = :input_sqc AND T4.t_account_id = T1.a_account_id;
```

Слика 4.4: Фаза претпроцесирања: кôд који садржи поглед (горе) и одговарајући кôд без њега (доле).

Синтакса подржаних SQL упита дата је граматиком приказаном на слици 4.5 (подразумевајући да су погледи већ елиминисани у фази претпроцесирања).

```
Query ::= SELECT Exprs
          FROM Tables
          [WHERE Cond]
          [ORDER BY Crit]
          [LIMIT num]
Exprs ::= Expr
        | Exprs, Expr
Expr  ::= num | table.column_name
        | :variable_name
        | Un_op Expr
        | Expr Bin_op Expr
        | function_name (Expr)
        | ( Expr )
Un_op ::= + | -
```

```
Bin_op     ::= + | - | * | % | /
Tables     ::= table
             | Tables, table
Cond       ::= Term
             | NOT Cond
             | Cond Binbool_op Cond
             | ( Cond )
Term       ::= TRUE | FALSE
             | Expr Bin_rel Expr
Bin_rel    ::= < | <= | ==
             | > | >= | <>
Binbool_op ::= AND | OR
Crit       ::= Expr [ASC | DESC]
             | Crit, Expr [ASC | DESC]
```

Слика 4.5: Подржана граматика SQL упита.

У C/C++ коду, за сваки IRsql упит Q уводе се две функције: E_Q и F_Q, чиме се добија међа-репрезентација IRc. Вредност функције E_Q говори да ли је било грешке при извршавању упита. Сам упит се мења једним или помоћу више позива функције F_Q: за сваку IRsql излазну променљиву v, уводи се додела облика `v = F_Q(...)` (са одговарајућим аргументима). Све променљиве које се користе као улазни аргументи IRsql упита Q су параметри обе функције. Неки додатни параметри могу бити уведени касније, што ће бити објашњено у поглављу 4.2.3.

Пример 4.5 Поделом функције са слике 4.2 (десно), добија се кôд представљен на слици 4.6.

```
SELECT T1.a_pending_blnc,
       T1.a_avail_blnc
  INTO :pending_blnc_sqc,
       :avail_blnc_sqc
FROM account AS T1
WHERE T1.a_account_id =
        :account_sqc
```

```
int pending_deposit(int account_id) {
  account_sqc = account_id;

  success_a         = E_Q(account_sqc);
  pending_blnc_sqc = F_Q(0, account_sqc);
  avail_blnc_sqc    = F_Q(1, account_sqc);

  if (!success_a) goto errorexit;

  return pending_blnc_sqc - avail_blnc_sqc;
}
```

Слика 4.6: Резултат поделе функције дате на слици 4.2 (десно): IRsql кôд (лево) и IRc кôд (десно).

Предложени приступ може се лако прилагодити и користити у различитим контекстима. После поделе на SQL и императивни кôд, даља анализа не зависи од оригиналног кода, већ само од међу-репрезнетација IRsql и IRc. Зато, да би се приступ користио у новом контексту, нпр. са ODBC API позивима уместо са уграђеним SQL-ом (пример еквивалентан са примером приказаним на слици 4.2 лево, дат је на слици 4.7), потребно је имплементирати другачије само фазу поделе, узимајући у обзир синтаксу језика и позива функција кроз API.

Ако матични језик није C/C++, онда алат LAV треба заменити одговарајућим проверивачем модела (енгл. *model checker*) – на пример, ако је матични језик *Java*, *JBMC* [62] може бити добар избор. Слично се могу анализирати и веб скрипт језици и приступ базама података из њих, нпр. PHP и MySql, или ASP.net и SQL Server.

```
int pending_deposit(int account_id) {
    account_sqc = account_id;

    sprintf (sqlstr, "SELECT T1.a_pending_blnc - T1.a_avail_blnc \
                      FROM account AS T1 \
                      WHERE T1.a_account_id = \%d", account_sqc);

    SQLExecDirect(hstmt, sqlstr, SQL_NTS);
    SQLBindCol(hstmt, 1, SQLINTEGER, &pending_deposit_sqc, 0, NULL);
    rc = SQLFetch(hstmt);

    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO) goto errorexit;

    return pending_deposit_sqc;
}
```

Слика 4.7: Пример функције која користи *Microsoft* ODBC интерфејс. Функција има еквивалентно понашање као функција представљена на слици 4.2 (лево).

4.2.2 Опис C/C++ функције: од IRc до FOL формуле

IRc кôд се анализира алатом LAV, који генерише FOLc формулу за опис свих могућих извршавања кроз анализирану функцију. Генерисана формула припада теорији **L** или **B**, које су уведене у поглављу 2.1. Сваки њен модел (ако се не користи уопштење полазног кода) одговара неком конкретном извршавању.

За ову сврху, алат LAV је незнатно допуњен. Позиви функција које су уведене због IRsql упита (функције $\mathtt{E}_Q$ и $\mathtt{F}_Q$) моделују се неинтерпретираним предикатима e и неинтерпретираним функцијама f. Моделовање својстава ових предиката и функција биће описано у поглављу 4.2.3 и управо та својства праве везу формуле FOLc (која представља IRc кôд) и скупа формула FOLsql (који репрезентује IRsql).

4.2.3 Опис SQL упита: од IRsql и схеме базе до FOL формуле

Својства новоуведених неинтерпретираних предиката и функција моделују се аксиомама. Њих генерише SQL2FOL алат који користи IRsql кôд и схему базе података (енгл. *database schema*) као улаз. У разматраној C/C++ функцији може бити више IRsql упита и сваки од њих се моделује појединачно. Ове аксиоме не моделују уграђени SQL потпуно прецизно, тј. користе се неке апстракције и упрошћења из разлога ефикасности.

Постоје две врсте C/C++ променљивих које се користе као веза између IRsql кода и самог кода у C/C++. Листа променљивих које се користе као улазни параметри упита обележава се са $\overrightarrow{in}$ и садржи n_{in} променљивих које се појављују у упиту. Листа променљивих које се користе као излаз упита обележава се са $\overrightarrow{out}$ и садржи n_{out} променљивих које узимају вредности из упита. Број n_{out} је једнак броју израза у оквиру SELECT клаузуле упита.

Сигнатура теорије

Сигнатура FOL теорије која се користи за моделовање IRsql упита над фиксираном базом података $\mathcal{D}$ је уређена петорка $(\tau, \mathcal{V}, \mathcal{F}, \mathcal{P}, \alpha)$, одређена скуповима:

- Типови: $\tau = \{tab,\ row,\ num,\ bool\}$. Интуитивно, tab одговара табелама, row редовима, num бројевима[1], а $bool$ одговара буловским вредностима.
- Пребројиви скуп типизираних променљивих, састављен од међусобно дисјунктних подскупова: $\mathcal{V} = \mathcal{V}_{tabs} \cup \mathcal{V}_{rows} \cup \mathcal{V}_{host} \cup \mathcal{V}_{bool}$.
- Функцијски симболи: $\mathcal{F} = \mathcal{N} \cup \mathcal{B} \cup \mathcal{A} \cup \mathcal{T} \cup \mathcal{C} \cup \mathcal{CP} \cup \{f, im, pos\}$, где:
  - $\mathcal{N}$ садржи нумеричке константе.
  - $\mathcal{B}$ садржи буловске константе $\top$ и $\bot$.
  - $\mathcal{A} = \mathcal{A}_{un} \cup \mathcal{A}_{bin}$ је скуп аритметичких функцијских симбола који се састоји од унарних $\mathcal{A}_{un} = \{+_{un}, -_{un}\}$[2] и бинарних симбола $\mathcal{A}_{bin} = \{+, -, *, mod, div\}$.
  - $\mathcal{T}$ садржи симболе константи именованих по именима свих табела у бази података $\mathcal{D}$.
  - $\mathcal{C}$ садржи симболе функција именованих по атрибутима свих табела у бази података $\mathcal{D}$, претпостављајући да су имена свих атрибута различита.[3]
  - $\mathcal{CP} = \cup_{n=2}^{\infty}\{cp_n\}$ је скуп функцијских симбола различите арности који одговарају спајању више табела (енгл. *cross products of tables*).
- Предикатски симболи: $\mathcal{P} = \mathcal{R} \cup \{e\}$, где:
  - $\mathcal{R} = \{=, <, \leq, >, \geq, \neq\}$ је скуп симбола релационих оператора.

и типизирајуће функције (енгл. *typing function*) α, дефинисане на следећи начин:

$$\alpha(v) = \begin{cases} tab, & \text{ако } v \in \mathcal{V}_{tables} \\ row, & \text{ако } v \in \mathcal{V}_{rows} \\ num, & \text{ако } v \in \mathcal{V}_{host} \\ bool, & \text{ако } v \in \mathcal{V}_{bool} \end{cases} \qquad \alpha(p) = \begin{cases} (num, ..., num), & \text{ако } p = e \\ (num, num), & \text{ако } p \in \mathcal{R} \end{cases}$$

$$\alpha(\phi) = \begin{cases} (num), & \text{ако } \phi \in \mathcal{N} \\ ((num), num), & \text{ако } \phi \in \mathcal{A}_{un} \\ (tab), & \text{ако } \phi \in \mathcal{T} \\ ((\underbrace{tab, \ldots, tab}_{n}), tab), & \text{ако } \phi = cp_n \\ ((num, ..., num), num), & \text{ако } \phi = f \end{cases} \quad \begin{cases} (bool), & \text{ако } \phi \in \mathcal{B} \\ ((num, num), num), & \text{ако } \phi \in \mathcal{A}_{bin} \\ ((row, tab), num), & \text{ако } \phi \in \mathcal{C} \\ ((row, tab), num), & \text{ако } \phi = pos \\ ((tab), tab), & \text{ако } \phi = im \end{cases}$$

[1] У овом тренутку, типови бројева нису прецизирани (осим претпоставке да за сваки број n важи $n \geq 0$ или $n < 0$). У имплементацији, овај тип се третира као тип целих бројева, или тип целобројних бит-вектора фиксне ширине. Дискусија на тему подршке других типова дата је у поглављу 5.2

[2] Унарни плус одговара унарном плусу у SQL-у, али може бити изостављен.

[3] У случају да имена атрибута у различитим табелама нису различита, на имена табела надовезују се имена атрибута, обезбеђујући јединствена имена неинтерпретираних функција.

Предикат e, функције f и pos, међу-табела $im(T)$ Постоје два различита исхода извршавања било ког SQL упита: тражени подаци су враћени из базе података, или се догодила нека грешка при извршавању (укључујући и случај када нема релевантних података у бази). У сврху моделовања оваквог понашања, за разматрани упит, користи се предикатски симбол e који одговара функцији $\mathtt{E}_\mathtt{Q}$ из IRsql-а и моделује постојање повратне вредности IRsql упита, као и функцијски симбол f, који одговара функцији $\mathtt{F}_\mathtt{Q}$ из IRsql и моделује резултујућу вредност, ако она постоји. Симбол e има арност:

- n_{in} у случају да је разматрани упит *s*-упит и
- $1 + n_{in}$ у случају *m*-упита. Овде се први аргумент користи за моделовање редног броја реда у *међу-табели* (енгл. *intermediate table*) $im(T)$ описаној ниже у тексту.

Симбол f има арност:

- n_{in} у случају *su*-упита и
- $1+n_{in}$ у случају *st*-упита. Овде први аргумент одговара стандарним SQL пројекцијама и њиме се контролише који елемент резултујуће енторке се враћа као резултат функције f.
- $1 + n_{in}$ у случају *mu*-упита. Први аргумент се користи као код предиката e.
- $2 + n_{in}$ у случају *mt*-упита. Његови аргументи су означени са $p, t, \overrightarrow{in}$, где се први користи као код предиката e, а други се користи на исти начин као први аргумент у случају *st*-упита.

Функцијски симболи im и pos уводе се због моделовања *m*-упита. Међу-табела $im(T)$ садржи оне редове оригиналне табеле T задате FROM клаузулом који задовољавају услове захтеване упитом (WHERE клаузулом). Ова табела не узима у обзир издвојене атрибуте и могуће трансформације над вредностима тих атрибута (они се моделују *Резултат* аксиомама). Интуитивно, неки ред може припадати и оригиналној таблели T и међу-табели $im(T)$, јер је она подскуп табеле T. Функција pos моделује редни број реда у табели. Симболи $im(T)$ и pos биће додатно разматрани у поглављу 4.3.

У следећим аксиомама (тј. шаблонима аксиома), T (или T_i) је константни симбол типа *tab* (а не квантификована променљива) који одговара табели T (или T_i) присутној у IRsql упиту. За атрибут табеле $\mathtt{T}_i$ именован $\mathtt{T}_i\_\mathrm{col}_j$, за $j \in [1..n_i]$, претпоставља се да је одговарајући функцијски симбол именован $T_i\_col_j$. У упитима, табеле се могу спајати, правећи нове табеле. Ако упит спаја табеле $\mathtt{T}_1$, ..., $\mathtt{T}_m$, тада се њихов производ означава са $cp(\mathtt{T}_1, \ldots, \mathtt{T}_m)$. Са con означен је услов из WHERE клаузуле, са $\mathtt{exp}_i$ за $i \in [1, \ldots n_{out}]$, изрази присутни у SELECT клаузули разматраног упита, са $\mathtt{crt}_i$ за $i \in [1, \ldots n_{crt}]$, изрази у ORDER BY клаузули, а lim означава константу у LIMIT клаузули.

Све аксиоме наведене у наставку претпостављају обраду над нумеричким вредностима. Овај избор је направљен да би се упростило моделовање, али се тиме не умањује моћ приступа [130].

Трансформисање услова и израза

Са циљем пресликавања израза и услова из IRsql-a у одговарајуће изразе и формуле из описане сигнатуре уводи се мета функција σ. Она служи као макро замена (енгл. *macro substitution*), па иако се јавља у нашим формулама, претпоставља се да је у потпуности развијена. Функција σ има три аргумента: први је типа *row*, други је типа *table*, а трећи може бити услов или израз из IRsql-a. У зависности од трећег аргумента `q`, ако је `q`:

- `true` или `false`, резултат је $\top$ или $\bot$ редом,
- нумеричка константа, резултат је одговарајућа константа из $\mathcal{N}$,
- променљива, резултат је одговарајућа променљива из $\mathcal{V}_{host}$,
- аритметички оператор, резултат је одговарајући оператор из $\mathcal{A}$,
- логички оператор, резултат је одговарајући логички везник у FOL формули,
- релациони оператор, резултат је одговарајући релациони оператор из $\mathcal{R}$,
- атрибут `c`, резултат је $c(r, T)$, где је c одговарајућа функција из $\mathcal{C}$,
- сложени израз или услов из IRsql, резулатат се добија рекурзивном применом претходних правила на његове аргументе и операторе.

Ако `WHERE` клаузула није присутна у IRsql упиту, онда је услов таквог упита празан (ε), и подразумева се да је $\sigma(r, T, \varepsilon)$ једнако $\top$.

Пример 4.6 Резултат примене функције σ на производ табела `account`, `transaction` и `customer`, и услов `t_txn_date < :in1 - 14` је:

$$\sigma\big(r, cp(\texttt{account}, \texttt{transaction}, \texttt{customer}), \texttt{t\_txn\_date} < \texttt{:in1 - 14}\big) =$$

$$t\_txn\_date\big(r, cp_3(account,\ transaction,\ customer)\big) < in1 - 14$$

Аксиоме

Аксиоме су приказане на слици 4.8. У зависности од типа упита и услова који треба да се провере (видети поглавље 4.3.2) само се одређене аксиоме користе. На пример, у случају *su*-упита, користе се само *s*- и *su*- аксиоме.

***Производ табела*:** Аксиома *Производ табела* за фиксиране табеле $\texttt{T}_i$ спаја вредности, атрибуте и редове производа табела са вредностима, атрибутима и редовима табела које се спајају. Упити обично не приступају вредностима свих атрибута неке табеле, већ само неком њиховом подскупу, па се аксиома *Производ табела* може упростити да укључује само оне атрибуте које се користе у оквиру упита.

Пример 4.7 Конкретна инстанца аксиоме *Производ табела* заснована на упиту из текућег примера (слика 4.1, лево) дата је на слици 4.9 (формула *Инстанца производа табела*).

$$\forall\, x_{1,1} : num,\ \ldots, x_{1,n_1} : num, \ldots, x_{m,n_m} : num, \qquad \textit{(Производ табела)}$$
$$\exists\, r : row \bigwedge_{i=1..m} \bigwedge_{j=1..n_i} T_i\_col_j(r, cp(T_1, \ldots, T_m)) = x_{i,j}$$
$$\Leftrightarrow \exists\, r_1 : row, ...,\ r_m : row \bigwedge_{i=1..m} \bigwedge_{j=1..n_i} T_i\_col_j(r_i, T_i) = x_{i,j}$$

$$cp(T_1, \ldots, T_m) = cp(T_{i_1}, \ldots, T_{i_m}) \qquad \textit{(Пермутација производа табела)}$$

$$\forall\, r_1 : row, r_2 : row \bigwedge_{i=1..n} T\_pk_i(r_1, T) = T\_pk_i(r_2, T)\ \Rightarrow\ r_1 = r_2 \qquad \textit{(Јединственост)}$$

$$\forall\, r_2 : row\ \exists r_1 : row \bigwedge_{i=1..n} T_2\_col_i(r_2, T_2) = T_1\_col_i(r_1, T_1) \qquad \textit{(Страни кључ)}$$

$$\forall\, \overrightarrow{in : num}\ (e(\overrightarrow{in}) \Leftrightarrow (\exists\, r : row\ \sigma(r, T, \texttt{con}) \wedge \qquad \textit{(s-Постојање)}$$
$$\forall\, r_1 : row, r_2 : row\ (\sigma(r_1, T, \texttt{con}) \wedge\ \sigma(r_2, T, \texttt{con}))\ \Rightarrow\ r_1 = r_2))$$

$$\forall\, \overrightarrow{in : num}, result : num \qquad \textit{(su-Резултат)}$$
$$e(\overrightarrow{in})\ \wedge f(\overrightarrow{in}) = result\ \Rightarrow \exists\, r : row\ (\sigma(r, T, \texttt{con})\ \wedge result = \sigma(r, T, \texttt{exp}))$$

$$\forall\, \overrightarrow{in : num}, result_1 : num, ..., result_{n_{out}} : num \qquad \textit{(st-Резултат)}$$
$$e(\overrightarrow{in}) \wedge \bigwedge_{t=1..n_{out}} f(t, \overrightarrow{in}) = result_t \Rightarrow \exists r : row(\sigma(r, T, \texttt{con}) \wedge \bigwedge_{t=1..n_{out}} result_t = \sigma(r, T, \texttt{exp}_t))$$

$$\forall \overrightarrow{in : num}, r : row \qquad \textit{(m-Позиција вредности)}$$
$$(\sigma(r, T, \texttt{con}) \Leftrightarrow (\exists p : num(p \geq 0 \wedge pos(r, im(T)) = p)))$$

$$\forall\, r_1 : row, r_2 : row, p : num \qquad \textit{(m-Позиција инјективности)}$$
$$(p \geq 0 \wedge pos(r_1, im(T)) = p \wedge pos(r_2, im(T)) = p \Rightarrow r_1 = r_2)$$

$$\forall\, \overrightarrow{in : num}, p : num\ (p \geq 0 \Rightarrow (e(p, \overrightarrow{in}) \Leftrightarrow \exists\, r : row\ pos(r, im(T)) = p)) \qquad \textit{(m-Постојање)}$$

$$\forall\, \overrightarrow{in : num}, result : num, p : num\ (p \geq 0 \wedge \qquad \textit{(mi-Резултат)}$$
$$e(p, \overrightarrow{in}) \wedge f(p, \overrightarrow{in}) = result \Rightarrow \exists\, r : row(pos(r, im(T)) = p \wedge result = \sigma(r, T, \texttt{exp})))$$

$$\forall\, \overrightarrow{in : num}, result_1 : num, ..., result_{n_{out}} : num, p : num\ (p \geq 0 \wedge e(p, \overrightarrow{in}) \wedge \qquad \textit{(mt-Резултат)}$$
$$\bigwedge_{t=1..n_{out}} f(p, t, \overrightarrow{in}) = result_t \Rightarrow \exists r : row(pos(r, im(T)) = p \wedge \bigwedge_{t=1..n_{out}} result_t = \sigma(r, T, \texttt{exp}_t)))$$

Слика 4.8: Аксиоме у вези са схемом базе података (прва група), аксиоме које одговарају s-упитима/m-упитима (друга/трећа група).

Пермутација производа табела: Производ табела садржи исте податке невезано за редослед спојених табела. Ово је моделовано аксиомом *Пермутација производа табела* (за свако m и сваку пермутацију $i_1, \ldots i_m$ низа $1, \ldots m$).

Пример 4.8 Конкретна инстанца аксиоме *Пермутација производа табела* заснована на упиту из текућег примера (слика 4.1, лево) дата је на слици 4.9 (формуле *Инстанце*

пермутација производа табела).

$$\forall\, x_0 : num,\ x_1 : num,\ x_2 : num,\ x_3 : num, \qquad \textit{(Инстанца производа табела)}$$
$$x_4 : num,\ x_5 : num,\ x_6 : num,\ x_7 : num,\ x_8 : num$$
$$\exists r : row\ (a\_account\_id(r,T) = x_0 \wedge a\_avail\_blnc(r,T) = x_1 \wedge a\_cust\_id(r,T) = x_2 \wedge$$
$$c\_cust\_id(r,T) = x_3 \wedge c\_cust\_type(r,T) = x_4 \wedge t\_account\_id(r,T) = x_5 \wedge$$
$$t\_amount(r,T) = x_6\ \wedge\ t\_txn\_date(r,T) = x_7 \wedge t\_txn\_id(r,T) = x_8\ )$$
$$\Leftrightarrow \exists\, r_1 : row,\ r_2 : row,\ r_3 : row$$
$$(a\_account\_id(r,A) = x_0 \wedge a\_avail\_blnc(r,A) = x_1 \wedge a\_cust\_id(r,A) = x_2 \wedge$$
$$c\_cust\_id(r,C) = x_3 \wedge c\_cust\_type(r,C) = x_4 \wedge t\_account\_id(r,N) = x_5 \wedge$$
$$t\_amount(r,N) = x_6 \wedge t\_txn\_date(r,N) = x_7 \wedge t\_txn\_id(r,N) = x_8\ )$$

$$cp_3(A,N,C) = cp_3(A,C,N)\ ; \qquad \textit{(Инстанце пермутација производа табела)}$$
$$cp_3(A,N,C) = cp_3(N,A,C)\ ;\ cp_3(A,N,C) = cp_3(N,C,A)\ ;$$
$$cp_3(A,N,C) = cp_3(C,A,N)\ ;\ cp_3(A,N,C) = cp_3(C,N,A)$$

$$\forall r_1{:}row, r_2{:}row\ (a\_account\_id(r_1,T){=}a\_account\_id(r_2,T) \qquad \textit{(Инстанца јединствености)}$$
$$\wedge\, t\_txn\_id(r_1,T) = t\_txn\_id(r_2,T) \wedge c\_cust\_id(r_1,T) = c\_cust\_id(r_2,T) \Rightarrow r_1 = r_2)$$

$$\forall\, r_2 : row\ \exists r_1 : row\ a\_cust\_id(r_2,A) = c\_cust\_id(r_2,C) \qquad \textit{(Инстанца страног кључа)}$$

$$\forall\, in_1 : num, in_2 : num,\ r : row \qquad \textit{(Инстанца m-позиције вредности)}$$
$$(\ t\_txn\_date(r,T) > in_1 - 14 \wedge \neg(a\_avail\_blnc(r,T) - in_2 \geq 0) \wedge$$
$$a\_cust\_id(r,T) = c\_cust\_id(r,T) \wedge a\_account\_id(r,T) = t\_account\_id(r,T)) \Leftrightarrow$$
$$(\exists p : num\ (p \geq 0 \wedge pos(r, im(T)) = p))$$

$$\forall\, r_1 : row, r_2 : row, p : num \qquad \textit{(Инстанца m-позиције инјективности)}$$
$$p \geq 0\ \wedge\ pos(r_1, im(T)) = p\ \wedge\ pos(r_2, im(T)) = p\ \Rightarrow\ r_1 = r_2$$

$$\forall\, in_1 : num, in_2 : num, p : num \qquad \textit{(Инстанца m-постојања)}$$
$$p \geq 0\ \Rightarrow\ (e(p, in_1, in_2) \Leftrightarrow \exists\, r : row\ pos(r, im(T)) = p)$$

$$\forall\, in_1 : num, in_2 : num, \qquad \textit{(Инстанца mt-резултата)}$$
$$rslt_1 : num, rslt_2 : num, rslt_3 : num, rslt_4 : num, rslt_5 : num, rslt_6 : num, p : num$$
$$p \geq 0 \wedge e(p, in_1, in_2) \wedge \bigwedge_{t=1..6} f(p, t, , in_1, in_2) = rslt_t \Rightarrow$$
$$\exists r : row\ (pos(r, im(T)) = p \wedge rslt_1 = a\_avail\_blnc(r,T) + t\_amount(r,T) \wedge$$
$$rslt_2 = in_1 - t\_txn\_date(r,T) \wedge rslt_3 = t\_txn\_id(r,T) \wedge$$
$$rslt_4 = a\_account\_id(r,T) \wedge rslt_5 = a\_cust\_id(r,T) \wedge rslt_6 = c\_cust\_type(r,T))$$

Слика 4.9: Инстанце аксиоми са слике 4.8 за упит са слике 4.1 (лево). Константе A, N и C означавају табеле *account*, *transaction* и *customer* редом, док константа T означава њихов производ, тј. $cp_3(A,\ N,\ C)$.

***Јединственост*:** У складу са стандардном семантиком примарног кључа [88], аксиома *Јединственост* описује својство јединствености примарних кључева свих табела поменутих у упиту, укључујући и табеле које се спајају. Информација о томе који је

атрибут примарни кључ, или од којих се атрибута састоји сложени примарни кључ, сачувана у схеми базе. Примарни кључ производа табела састоји се од примарних кључева табела које се спајају. Својство јединствености (дато кључном речју `UNIQUE` у схеми базе података) може постојати и за друге скупове атрибута, па се ова аксиома уводи и за њих.

Пример 4.9 Претпостављајући да свака од три табеле из текућег примера (слика 4.1) има тачно један атрибут као примарни кључ, ред из њиховог производа биће јединствено одређен комбинацијом тих тријy кључева, како је и представљено на слици 4.9 (формула *Инстанца јединствености*).

***Страни кључ*:** Постоје важне везе између табела које су наметнуте дизајном базе података. На пример, нека за свако $i \in [1, \ldots, n]$ атрибути $\texttt{T}_1\texttt{\_col}_i$ представљају сложени примарни кључ табеле $\texttt{T}_1$, и нека је $\texttt{T}_2\texttt{\_col}_i$ сложени страни кључ табеле $\texttt{T}_2$ такав да ти атрибути задовољавају везу примарни-страни кључ. То се моделује аксиомом *Страни кључ*.

Пример 4.10 Инстанца аксиоме *Страни кључ* за текући пример (слика 4.1), представљена је на слици 4.9 (формула *Инстанца страног кључа*).

У следећем тексту и аксиомама које зависе од упита (*упит-зависне* аксиоме), T означава једну табелу (ако је само она присутна у упиту), или производ табела $cp(T_1, ...T_m)$ (ако се табеле $\texttt{T}_i$ спајају у упиту).

***s-Постојање*:** Аксима *s-Постојање* моделује да упит враћа релевантне јединствене податке из базе ако и само ако постоји јединствени ред r такав да је услов упита задовољен вредностима реда r.

Пример 4.11 Конкретна инстанца аксиоме *s-Постојање* заснована на упиту са слике 4.2 приказана је на слици 4.10 (формула *Инстанца s-постојања*).

$$\forall\, in : num\ (e(in) \Leftrightarrow (\exists\, r : row\ a\_account\_id(r, A) = in \wedge \quad \text{(Инстанца s-постојања)}$$
$$\forall\, r_1 : row,\ r_2 : row\ (a\_account\_id(r_1, A) = in \wedge$$
$$a\_account\_id(r_2, A) = in)\ \Rightarrow\ r_1 = r_2))$$

$$\forall\, in : num, rslt_1 : num, rslt_2 : num \quad \text{(Инстанца st-резултата)}$$
$$e(in) \wedge f(0, in) = rslt_1 \wedge f(1, in) = rslt_2 \Rightarrow$$
$$\exists r : row\ (a\_account\_id(r, A) = in\ \wedge\ rslt_1 = a\_pending\_blnc(r, A) \wedge$$
$$rslt_2 = a\_avail\_blnc(r, A))$$

Слика 4.10: Инстанце аксиоми представљених на слици 4.8 за упит са слике 4.2 (десно). Константа A означава табелу *account*.

***m*-Позиција вредност, *m*-Позиција инјективност:** Аксиоме *m-Позиција вредност* и *m-Позиција инјективност* описују нека проста својства функције *pos*: за редове који задовољавају услов дат упитом, она враћа ненегативне вредности и функција је инјективна.

Пример 4.12 Инстанце аксиома *m-Позиција вредност* и *m-Позиција инјективност* базиране на текућем примеру (слика 4.1, лево) приказане су на слици 4.9 (формуле *Инстанца m-позиције вредности* и *Инстанца m-позиције инјективности*, редом).

***m*-Постојање:** Аксиома *m-Постојање* користи се да повеже постојање редова који задовољавају дати услов са њиховим позицијама.

Пример 4.13 За текући пример (слика 4.1, лево), одговарајућа инстанца аксиоме *m-Постојање* дата је на слици 4.9 (формула *Инстанца m-постојања*).

***su*-Резултат, *st*-Резултат, *mu*-Резултат, *mt*-Резултат:** Вредности враћене извршавањем упита моделују се *Резултат* аксиомама које одговарају типу упита, у зависности да ли је упит *su*-, *st*-, *mu*- или *mt*-упит.

Пример 4.14 Конкретна инстанца аксиоме *st-Резултат*, добијена на примеру упита приказаног на слици 4.2 (десно), приказана је на слици 4.10 (формула *Инстанца st-резултата*). Инстанца аксиоме *mt-Резултат*, која одговара текућем примеру (4.1, лево) дата је на слици 4.9 (формула *Инстанца mt-резултата*).

4.2.4 Конструкција описа функције

Основни опис функције са уграђеним SQL-ом је скуп формула који укључује FOLc формулу и скуп аксиома FOLsql. Аксиоме FOLsql (за одговарајуће табеле и атрибуте) описују IRsql упите за све могуће вредности улаза у упит и излаза из њега. На пример, ако са Ax означимо аксиому, и ако са $\overrightarrow{sql\_v}$ означимо све променљиве које се јављају у упиту, онда општи облик аксиоме можемо записати у наредном облику:

$$\forall \overrightarrow{sql\_v}\ Ax(\overrightarrow{sql\_v}).$$

Овако генерисана аксиома може се користити за сваку функцију са оваквим уграђеним упитом. Међутим, за даље доказивање потребне су само инстанце ових аксиома за конкретне вредности променљивих $\overrightarrow{sql\_v}$ које се појављују у разматраном коду. Уколико су променљиве у матичном језику означене са $\overrightarrow{host\_v}$, тада је потребно користити само инстанцу аксиоме која каже да услови аксиоме важе за те конкретне променљиве, тј.

$$Ax(\overrightarrow{host\_v}).$$

Пример 4.15 У аксиоми *Инстанца mt-резултата* са слике 4.9, све променљиве in_1, in_2, $rslt_1, ..., rslt_6$ могу бити инстанциране одговарајућим вредностима из FOLc.

Такође, аксиоме за *m*-упите користе се када има промена у коду после упита, и када се испитује k-еквиваленција (видети поглавље 4.3.2). У таквим случајевима, уместо аксиома универзално квантификованих по p, користе се одговарајуће инстанце за вредности променљиве p од 0 до k. Ово важи за аксиоме *Инстанца m-позиције вредности*, *Инстанца m-позиције инјективности* и *Инстанца mt-резултата* са слике 4.9.

Пример 4.16 За аксиому *Инстанца т-позиције инјективности*, одговарајуће конкретизоване инстанце дате су на слици 4.11.

$$\forall\, r_1 : row, r_2 : row\ (0 \geq 0 \wedge pos(r_1, im(T)) = 0 \wedge pos(r_2, im(T)) = 0\ \Rightarrow\ r_1 = r_2) \qquad (p = 0)$$

$$\forall\, r_1 : row, r_2 : row\ (1 \geq 0 \wedge pos(r_1, im(T)) = 1 \wedge pos(r_2, im(T)) = 1\ \Rightarrow\ r_1 = r_2) \qquad (p = 1)$$

$$\forall\, r_1 : row, r_2 : row\ (2 \geq 0 \wedge pos(r_1, im(T)) = 2 \wedge pos(r_2, im(T)) = 2\ \Rightarrow\ r_1 = r_2) \qquad (p = 2)$$

$$\forall\, r_1 : row, r_2 : row\ (3 \geq 0 \wedge pos(r_1, im(T)) = 3 \wedge pos(r_2, im(T)) = 3\ \Rightarrow\ r_1 = r_2) \qquad (p = 3)$$

$$\forall\, r_1 : row, r_2 : row\ (4 \geq 0 \wedge pos(r_1, im(T)) = 4 \wedge pos(r_2, im(T)) = 4\ \Rightarrow\ r_1 = r_2) \qquad (p = 4)$$

Слика 4.11: Инстанце аксиоме *Инстанца т-позиције инјективности* представљене на слици 4.9. Константа T означава производ $cp_3(account,\ transaction,\ customer)$.

Коришћењем таквих инстанци аксиома знатно се убрзава процес доказивања.

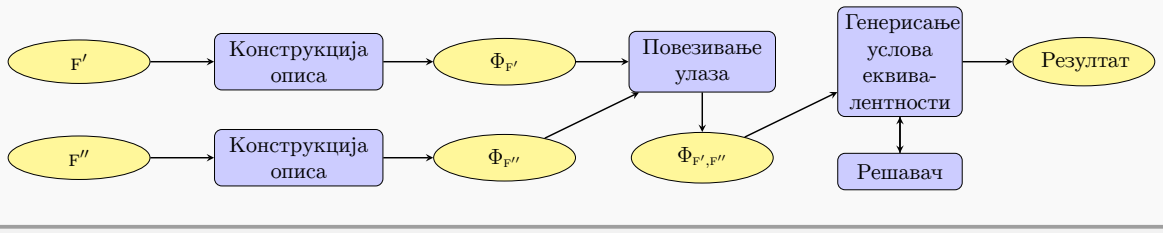
Опис функције F, конјункција формуле FOLc и формула FOLsql инстанцираних на приказан начин, конструише се као формула $\Phi_{\mathrm{F}}(\overrightarrow{input})$ где $\overrightarrow{input}$ означава улазне вредности функције F. Треба приметити да су све променљиве $\overrightarrow{host\_v}$ зависне од променљивих $\overrightarrow{input}$.

4.3 Конструкција услова еквивалентности

Конструисање услова који описује еквивалентност између два програма је захтеван задатак. У предложеном моделовању, претпоставка је да се провера еквивалентности ради у контексту рефакторисања, тј. када се две верзије исте функције пореде (у оквиру непромењеног остатка програма).

Приказ предложене архитектуре SQLAV алата дат је на слици 4.12. Почевши од две функције F' и F'', конструишу се њихови модели на начин описан у поглављу 4.2. Прецизније, за сваки анализиран упит понаособ, уводе се одговарајући њему специфични симболи f, e, im и pos, као и одговарајуће аксиоме, тј. резултујућа сигнатура се проширује да садржи различите симболе f, e, im и pos за сваки појединачни упит. На пример, када се анализира еквивалентност два m-упита, уводе се симболи pos' и pos'', и аксиома *т-Позиција вредности* ће бити конструисана и за pos' и за pos''. У контексту доказивања функционалне еквивалентности, подразумева се да су обе функције покренуте са истим улазим вредностима (и бројеви ових улазних вредности су исти), па формула $\Phi_{\mathrm{F}',\mathrm{F}''}$ која описује оба извршавања за фиксирани улаз $\overrightarrow{input}$ дефинише се као $\Phi_{\mathrm{F}'}(\overrightarrow{input}) \wedge \Phi_{\mathrm{F}''}(\overrightarrow{input})$. Ако треба проверити неко својство Ψ односа између ове две функције, проверава се ваљаност формуле $\Phi_{\mathrm{F}',\mathrm{F}''} \implies \Psi$ тј. $\Phi_{\mathrm{F}',\mathrm{F}''} \wedge \neg\Psi \implies \bot$ у одговарајућој теорији. Као основна теорија, може бити изабрана $\mathbf{L}$ или $\mathbf{B}$, проширена додатним аксиомама при формирању формуле $\Phi_{\mathrm{F}',\mathrm{F}''} \wedge \neg\Psi$. SMT решавачи какав је *Z3* могу резоновати са конјункцијама овог типа. Ове проширене теорије (енгл. *augmented theories*) биће означене са $\mathbf{L}^a$ и $\mathbf{B}^a$.

Својство Ψ одговараће различитим врстама услова еквивалентности, укључујући *услове еквиваленције у матичном језику* (енгл. *host language equivalence conditions* –



Слика 4.12: Приказ предложене архитектуре алата.

hlC) и *упит-зависне услове еквиваленције* (енгл. *query-specific equivalence conditions* – *qsC*). Конструкција свих тих услова и предложени алгоритам (слика 4.16) дискутују се у тексту који следи.

hlC услови конструишу се за уграђене *s*-упите и *m*-упите. Ови услови проверавају да ли су:

- повратне вредности функција у C/C++ једнаке,
- обе функције извршавају макрое за обраду грешака у еквивалентним случајевима, и
- глобалне променљиве и променљиве које се мењају преко показивача су постављене на исте вредности.

Прецизније, за *m*-упите се претпоставља да се вредности прочитане из базе уписују у низове пре њиховог даљег коришћења, па услови *hlC* садрже провере да ли одговарајући низови имају исти садржај (до на разматрану вредност k).

Пример 4.17 За упите представљене на слици 4.2, услов *hlC* треба да провери да ли су повратне вредности тих функција исте или обе завршавају скоком на `errorexit` лабелу.

Пример 4.18 За текући пример (слика 4.1), услов *hlC* проверава да ли су низови који садрже добијене вредности из базе исти у дужини и садржају.

Ако у анализираним функцијама постоје позиви екстерних функција, тј. оних за које не постоји доступна дефиниција, еквиваленција се не може доказати. Кôд функције мора бити доступан да би одговарајуће моделовање дозвољавало разматрање постојања релације еквиваленције.

4.3.1 Провера еквивалентности уграђених *s*-упита

Еквивалентност функција са уграђеним *s*-упитима моделује се условима $hl\mathcal{C}$. У овом случају, петље се ретко јављају (јер се не користе курсори), па се еквивалентност може доказати. Ипак, у случају присуства петљи без горње границе (или са високом границом), (само) k-еквивалентност (за неко конкретно k) се може доказати. Приступ се може користити за одређивање еквивалентности две функције са различитим бројем *s*-упита.

Пример 4.19 Неки брзи увиди у табелу (енгл. *short lookups*) могу се искомбиновати у један упит (слика 4.13) коришћењем аксиоме *Јединствености*.

```
...
input_sqc = a;
EXEC SQL SELECT t_amount
 INTO :amount_sqc
 FROM transaction
 WHERE t_txn_id = :input_sqc;
...
EXEC SQL SELECT t_account_id
 INTO :account_id_sqc
 FROM transaction
 WHERE t_txn_id = :input_sqc;
...
EXEC SQL SELECT a_avail_blnc
 INTO :avail_blnc_sqc
 FROM account
 WHERE a_account_id = :account_id_sqc;
...
return avail_blnc_sqc - amount_sqc;
```

```
...
input_sqc = a;
EXEC SQL SELECT a_avail_blnc
                - t_amount
           INTO :new_blnc_sqc
 FROM transaction, account
 WHERE t_txn_id = :input_sqc
   AND t_account_id =
       a_account_id;
...
return new_blnc_sqc;
```

Слика 4.13: Брзи увиди у табелу (лево), и спајање табела (десно)

Пример 4.20 Пример са слике 4.14 илуструје да се приступ може користити да закључи да су нека спајања и филтери непотребни користећи аксиому *Страни кључ*.

```
SELECT a_account_id
  INTO :account_id_sqc
FROM account, transaction
WHERE t_txn_id = :txn_id_sqc
  AND a_account_id = t_account_id
```

```
SELECT t_account_id
  INTO :account_id_sqc
FROM transaction
WHERE t_txn_id = :txn_id_sqc
```

Слика 4.14: Страни кључ `t_account_id` табеле `transaction` прави спајање табела непотребним.

4.3.2 Провера еквивалентности уграђених m-упита

Предложено моделовање може се користити за проверу еквивалентности када кôд садржи један m-упит. Оваква претпоставка је задовољавајућа у многим (ако не и свим) реалним ситуацијама. У овом случају, конструишу се и услови $hl\mathcal{C}$ (на већ описан начин), и услови $qs\mathcal{C}$ приказани на слици 4.15. Међутим, не проверавају се увек сви они. На пример, услови дати у кораку (m7) алгоритма приказаног на слици 4.16 проверавају се само у случају да се функције не разликују у коду после упита. У зависности од формуле која се докаже или оповргне, одговарајућа акција се предузима. На пример, ако услови из корака (m2) или (m3) алгоритма не важе, анализа се прекида уз резултат да еквиваленција не може бити доказана. У супротном, ако услови из корака (m3) важе, додаје се нова аксиома (оправдана доказаним условом) у формулу $\Phi_{\mathrm{F}',\mathrm{F}''}$. Детаљи о условима и одговарајућим акцијама дати су у тексту који следи, и илустровани примером са слике 4.1. За доказивање услова $qs\mathcal{C}$, није неопходно разматрање формуле $\Phi_{\mathrm{F}',\mathrm{F}''}$ у целини, већ је довољан само онај део који описује кôд пре уграђених SQL упита. Тај део ће бити означен са $\Phi_{\bar{\mathrm{F}}',\bar{\mathrm{F}}''}$.

$$\forall\, r_1 : row, r_2 : row\ (\sigma(r_1, T, \mathtt{con}) \wedge \sigma(r_2, T, \mathtt{con}) \Rightarrow \quad \textit{(Услов детерминистичког поретка)}$$
$$(\bigwedge_{t=1..n_{\mathtt{crt}}} \sigma(r_1, T, \mathtt{crt}_t) = \sigma(r_2, T, \mathtt{crt}_t) \Rightarrow \bigwedge_{t=1..n_{out}} \sigma(r_1, T, \mathtt{exp}_t) = \sigma(r_2, T, \mathtt{exp}_t)))$$

$$T' = T'' \qquad \textit{(Услов спајања)}$$

$$\forall\, r : row\ (\sigma(r, T', \mathtt{con}') \Leftrightarrow \sigma(r, T'', \mathtt{con}'')) \qquad \textit{(Услов филтрирања)}$$

$$\forall\, r_1 : row, r_2 : row\ (\sigma(r_1, T', \mathtt{con}') \wedge \sigma(r_2, T'', \mathtt{con}'') \Rightarrow \quad \textit{(Услов поретка)}$$
$$((\sigma(r_1, T', \mathtt{crt}'_1), \ldots, \sigma(r_1, T', \mathtt{crt}'_{n'_{\mathtt{crt}}})) \preceq' (\sigma(r_2, T', \mathtt{crt}'_1), \ldots, \sigma(r_2, T', \mathtt{crt}'_{n'_{\mathtt{crt}}})) \Leftrightarrow$$
$$(\sigma(r_1, T'', \mathtt{crt}''_1), \ldots, \sigma(r_1, T'', \mathtt{crt}''_{n''_{\mathtt{crt}}})) \preceq'' (\sigma(r_2, T'', \mathtt{crt}''_1), \ldots, \sigma(r_2, T'', \mathtt{crt}''_{n''_{\mathtt{crt}}}))))$$

$$\forall\, r : row\ (n'_{out} = n''_{out} \ \wedge\ \sigma(r, T', \mathtt{con}') \wedge \sigma(r, T'', \mathtt{con}'') \Rightarrow \quad \textit{(Услов пројекција)}$$
$$\bigwedge_{t=1..n'_{out}} \sigma(r, T', \mathtt{exp}'_t) = \sigma(r, T'', \mathtt{exp}''_t)\ )$$

Слика 4.15: Упит-зависни услови еквиваленције.

Кôд анализиране функције се дели у три дела: кôд пре уграђеног SQL упита, SQL упит и кôд после њега. Промене (тј. разлике у двема функцијама) могуће су у било ком делу, или у комбинацији било којих делова. Промене које су присутне искључиво у C/C++ делу, или само у уграђеном SQL упиту могу и треба да буду обрађени алатима специјализованим за рефакторисање у C/C++ или SQL-у. Фокус овог истраживања је на променама које захватају оба језика, и C/C++ и SQL, и где се не може доказати еквивалентност посматрајући их одвојено.

Ако не постоје промене анализиране функције пре упита, онда су делови описа F' и F'' који се односе на кôд пре упита исти. Зато се ради убрзавања процеса доказивања формули $\Phi_{\bar{\mathrm{F}}',\bar{\mathrm{F}}''}$ додаје формула која експлицитно каже да су променљиве које се користе као улазне у упит исте.

(s1)
```
if s-query
        try to prove ∀ input⃗ (Φ_{F′,F″} ⇒ hlC)
        if failed return cannot prove equivalence
        if loops were present return k-equivalent
        return equivalent
```

(m1)
```
if no changes before the SQL code
        add host_in_sql′⃗ = host_in_sql″⃗ into Φ_{F′,F″}, Φ_{F̄′,F̄″}
```

(m2)
```
try to prove ∀ input⃗ (Φ_{F̄′,F̄″} ⇒ jC)
if failed
        print "Tables are not equivalent!"
        return cannot prove equivalence
add (T′ = T″) into Φ_{F′,F″}, Φ_{F̄′,F̄″}
```

(m3)
```
try to prove ∀ input⃗ (Φ_{F̄′,F̄″} ⇒ fC)
if failed
        print "Filters are not equivalent!"
        return cannot prove equivalence
add (im′(T′) = im″(T″)) into Φ_{F′,F″}
add Kodomen pozicionih funkcija into Φ_{F′,F″}
```

(m4)
```
try to prove ∀ input⃗ (Φ_{F̄′} ⇒ dC′), ∀ input⃗ (Φ_{F̄″} ⇒ dC″)
if failed
        print "Orderings are not deterministic!"
        append primary keys to orderings
        or append expressions to make sel-orderings
```

(m5)
```
try to prove ∀ input⃗ (Φ_{F̄′,F̄″} ⇒ oC)
if failed print "Fix orderings!"
if sel-ord. add Jednakost kriterijuma pozicionih funkcija into Φ_{F′,F″}
else add Jednakost pozicionih funkcija into Φ_{F′,F″}
```

(m6)
```
if lim′ ≠ lim″ print "Fix limits!"
```

(m7)
```
if no changes after SQL code
        try to prove ∀ input⃗ (Φ_{F̄′,F̄″} ⇒ sC)
        if failed return cannot prove equivalence
        if loops were present before loop return k-equivalent
        return equivalent
```

(m8)
```
try to prove ∀ input⃗ (Φ_{F′,F″} ⇒ hlC)
if failed return cannot prove equivalence
if loops were present return k-equivalent
return equivalent
```

Слика 4.16: Алгоритам за проверу услова еквиваленције: корак (s1) за *s*-упите и кораци (m1)–(m8) за *m*-упите.

Пример 4.21 За текући пример са слике 4.1 постоје промене анализиране функције пре упита, па се у кораку (m1) ништа не додаје.

Важан циљ у доказивању је провера да ли су међу-табеле ових упита исте. Ако то не важи, еквиваленција кода се не може доказати предложеним моделовањем.[4] Међу-

[4]Еквиваленција је могућа ако, на пример, једна функција има део уобичајеног филтера имплемен-

табеле се разматрају јер, у еквивалентним кодовима, вредности у издвојеним редовима из две резултујуће табеле одговарајућих упита не морају бити исте, тј. могу бити промењене на различите начине у IRsql коду, па се затим та разлика може надокнадити трансформацијама у оквиру IRc кода. На пример, издвојена вредност може бити `x` у првом и `x+1` у другом упиту, док се у првом случају издвојена вредност увећава за 1 у C/C++, а у другом не. Еквиваленција међу-табела проверава се условима *Услов спајања* и *Услов филтрирања*.

Услов спајања (jC): Клаузула `FROM` може садржати једну или више табела, тј. T' и T'' (из два упита) могу бити просте табеле, или њихови производи. Овде се разматра најчешћа ситуација када су те табеле исте (потпуно исте табеле, или производи истих табела који се разликују евентуално до на редослед табела). У општем случају, међутабеле не морају нужно садржати исте редове ако се редови узимају из различитих, независних појединачних табела (под независним табелама подразумевају се табеле без било каквих додатно наметнутих услова на њихов међусобни однос). У неким специјалним случајевима, могуће је да различите табеле са истим типовима колона садрже исте редове, али за потребе доказивање еквиваленције неопходно је да ово важи за сваке две инстанце ових табела, што у општем случају није обавезно. Ако су табеле зависне од неке заједничке табеле, онда резултати могу бити исти (у неким специјалним, вештачким случајевима). Зависности табела и рефакторисање у овим случајевима може се разматрати само на SQL нивоу, и тада треба применити SQL засноване приступе рефакторисања (јер овај тип рефакторисања није повезан са променама у императивном делу кода). Ако су T' и T'' табеле, онда услов jC одговара провери да ли ради о истим константама и ово се тривијално проверава (јер су погледи елиминисани у претпроцесирању). У случају спајања табела, у провери еквиваленције табела користи се аксиома *Пермутација производа табела*.

Пример 4.22 За пример дат на слици 4.1 одговарајућа инстанца услова *Услов спајања* приказана је на слици 4.17 (формула *Инстанца услова спајања*).

Ако је овај услов задовољен, одговарајућа једнакост табела додаје се у формулу да би се могла користити у резоновањима која следе (ово одговара кораку (m2) у алгоритму).

Услов филтрирања (fC): Претпостављајући да је услов jC задовољен, међу-табеле садрже исте редове ако је задовољен и услов fC. Дакле, ако су оба услова задовољена међу-табеле су исте, па се у даљем резоновању користи формула $\Phi_{F',F''} \wedge (im'(T') = im''(T''))$.

Пример 4.23 За пример са слике 4.1, услов fC приказан је на слици 4.17 (формула *Инстанца услова филтрирања*). Како је он задовољен, једнакост $im'(cp_3(A, N, C)) = im''(cp_3(C, A, N))$ се додаје формули која ће се користити у резоновањима која следе (ово одговара кораку (m3) у алгоритму). Треба приметити да је услов у левом упиту `t_txn_date > :in1_sqc - 14` еквивалентан услову `t_txn_date > :in1_sqc` на десној страни, јер је тада вредност `:in1_sqc` већ смањена за 14 у C/C++ језику. На пример, ако би у услову `a_avail_blnc < :in2_sqc` на десној страни уместо оператора $<$ био употребљен оператор $\leq$, услов fC не би био задовољен.

тиран у C/C++ уместо у самом упиту. Међутим, такве ситуације су ретке, јер се филтрирање обично дешава као део SQL упита.

$\forall\, r_1 : row, r_2 : row$ (*Инстанца услова детерминистичког поретка*)

$$
\begin{aligned}
&(\ t\_txn\_date(r_1,T) > in_1 - 14\ \wedge \neg(a\_avail\_blnc(r_1,T) - in_2 \geq 0) \wedge \\
&\ \ a\_cust\_id(r_1,T) = c\_cust\_id(r_1,T)\ \wedge a\_account\_id(r_1,T) = t\_account\_id(r_1,T)) \wedge \\
&(\ t\_txn\_date(r_2,T) > in_1 - 14\ \wedge \neg(a\_avail\_blnc(r_2,T) - in_2 \geq 0) \wedge \\
&\ \ a\_cust\_id(r_2,T) = c\_cust\_id(r_2,T)\ \wedge a\_account\_id(r_2,T) = t\_account\_id(r_2,T))\ \Rightarrow \\
&((\ -t\_amount(r_1,T) = -t\_amount(r_2,T) \wedge \\
&\quad a\_avail\_blnc(r_1,T) + t\_amount(r_1,T) = a\_avail\_blnc(r_2,T) + t\_amount(r_2,T) \wedge \\
&\quad in_1 - t\_txn\_date(r_1,T) = in_1 - t\_txn\_date(r_2,T) \wedge \\
&\quad a\_cust\_id(r_1,T) = a\_cust\_id(r_2,T)\ \wedge\ a\_account\_id(r_1,T) = a\_account\_id(r_2,T) \wedge \\
&\quad t\_txn\_id(r_1,T) = t\_txn\_id(r_2,T))\ \Rightarrow \\
&\ \ (a\_avail\_blnc(r_1,T) + t\_amount(r_1,T) = a\_avail\_blnc(r_2,T) + t\_amount(r_2,T) \wedge \\
&\quad in_1 - t\_txn\_date(r_1,T) = in_1 - t\_txn\_date(r_2,T) \wedge\ t\_txn\_id(r_1,T) = t\_txn\_id(r_2,T) \wedge \\
&\quad a\_account\_id(r_1,T) = a\_account\_id(r_2,T)\ \wedge\ a\_cust\_id(r_1,T) = a\_cust\_id(r_2,T) \wedge \\
&\quad c\_cust\_type(r_1,T) = c\_cust\_type(r_2,T)))
\end{aligned}
$$

$T = T'$ (*Инстанца услова спајања*)

$\forall\, r : row$ (*Инстанца услова филтрирања*)

$$
\begin{aligned}
&(\ t\_txn\_date(r,T) > in_1 - 14\ \wedge \neg(a\_avail\_blnc(r,T) - in_2 \geq 0) \wedge \\
&\ \ a\_cust\_id(r,T) = c\_cust\_id(r,T)\ \wedge\ a\_account\_id(r,T) = t\_account\_id(r,T)) \Leftrightarrow \\
&(\ t\_txn\_date(r,T') > in'_1\ \wedge a\_avail\_blnc(r,T') < in'_2 \wedge \\
&\ \ a\_cust\_id(r,T') = c\_cust\_id(r,T')\ \wedge\ a\_account\_id(r,T') = t\_account\_id(r,T'))
\end{aligned}
$$

$\forall\, r_1 : row, r_2 : row$ (*Инстанца услова поретка*)

$$
\begin{aligned}
&(\ t\_txn\_date(r,T) > in_1 - 14\ \wedge \neg(a\_avail\_blnc(r,T) - in_2 \geq 0) \wedge \\
&\ \ a\_cust\_id(r,T) = c\_cust\_id(r,T)\ \wedge\ a\_account\_id(r,T) = t\_account\_id(r,T)) \wedge \\
&(\ t\_txn\_date(r,T') > in'_1\ \wedge a\_avail\_blnc(r,T') < in'_2 \wedge \\
&\ \ a\_cust\_id(r,T') = c\_cust\_id(r,T')\ \wedge\ a\_account\_id(r,T') = t\_account\_id(r,T'))\ \Rightarrow \\
&(((-t\_amount(r_1,T),\ a\_avail\_blnc(r_1,T) + t\_amount(r_1,T),\ in_1 - t\_txn\_date(r_1,T), \\
&\quad a\_cust\_id(r_1,T),\ a\_account\_id(r_1,T),\ t\_txn\_id(r_1,T)) \preceq' \\
&\ \ (-t\_amount(r_2,T),\ a\_avail\_blnc(r_2,T) + t\_amount(r_2,T),\ in_1 - t\_txn\_date(r_2,T), \\
&\quad a\_cust\_id(r_2,T),\ a\_account\_id(r_2,T),\ t\_txn\_id(r_2,T))) \Leftrightarrow \\
&\ \ ((-t\_amount(r_1,T'),\ a\_avail\_blnc(r_1,T'),\ t\_txn\_date(r_1,T'), \\
&\quad c\_cust\_id(r_1,T'),\ a\_account\_id(r_1,T'),\ t\_txn\_id(r_1,T')) \preceq' \\
&\ \ (-t\_amount(r_2,T'),\ a\_avail\_blnc(r_2,T'),\ -t\_txn\_date(r_2,T'), \\
&\quad c\_cust\_id(r_2,T'),\ a\_account\_id(r_2,T'),\ t\_txn\_id(r_2,T'))))
\end{aligned}
$$

Слика 4.17: Инстанце услова са слике 4.15 за кодове са слике 4.1. Константа T означава производ $cp_3(account,\ transaction,\ customer)$, док константа T' означава $cp_3(transaction,\ customer,\ account)$. Променљиве in_1 и in_2 одговарају улазним променљивама за SQL упит са слике 4.1 (лево), док in'_1 и in'_2 одговарају његовој рефакторисаној верзији.

Две међу-табеле су сортиране у поретку задатом клаузулом ORDER BY, и одговарајући поредак редова моделује се позиционим функцијама pos' и pos''. За свака два поретка у истој табели, може се претпоставити да позиционе функције имају исти кодомен. Ово је моделовано аксиомом *Кодомен позиционих функција* (слика 4.18), која се додаје формули $\Phi_{\mathrm{F}',\mathrm{F}''}$ у оквиру корака (m3).

$$\forall\, p : num\ (p \geq 0 \Rightarrow \qquad \textit{(Кодомен позиционих функција)}$$
$$(\exists\, r : row\ pos'(r, im'(T')) = p \Leftrightarrow\ \exists\, r : row\ pos''(r, im''(T'')) = p))$$

$$\forall\, r : row\ (pos'(r, im'(T')) = pos''(r, im''(T''))) \qquad \textit{(Једнакост позиционих функција)}$$

$$\forall\, r_1 : row, r_2 : row,\ p : num\ (p \geq 0 \Rightarrow \qquad \textit{(Једнакост критеријума позиционих функција)}$$
$$((pos'(r_1, im'(T')) = p \wedge pos''(r_2, im''(T'')) = p) \Rightarrow$$
$$\bigwedge_{i=1..n'_{\mathtt{crt}}} \sigma(r_1, T', \mathtt{crt}'_i) = \sigma(r_2, T', \mathtt{crt}'_i) \wedge \bigwedge_{i=1..n''_{\mathtt{crt}}} \sigma(r_1, T'', \mathtt{crt}''_i) = \sigma(r_2, T'', \mathtt{crt}''_i)))$$

Слика 4.18: Аксиоме позиционих функција.

Пример 4.24 За пример са слике 4.1, инстанца аксиоме *Кодомен позиционих функција* приказана је на слици 4.19 (формула *Инстанца кодомена позиционих фја*).

$$\forall\, p : num\ (p \geq 0 \Rightarrow \qquad \textit{(Инстанца кодомена позиционих фја)}$$
$$(\exists\, r : row\ pos'(r, im'(cp_3(A, N, C))) = p \Leftrightarrow \exists\, r : row\ pos''(r, im''(cp_3(C, A, N))) = p))$$

$$\forall\, r : row \qquad \textit{(Инстанца једнакости позиционих функција)}$$
$$pos'(r, im'(cp_3(A, N, C))) = pos''(r, im''(cp_3(C, A, N)))$$

Слика 4.19: Инстанце аксиома позиционих функција са слике 4.18, за кôд приказан на слици 4.1

Услов детерминистичког поретка (dC): Поредак је битан јер одређује у ком ће се редоследу редови појавити у курсору и бити коришћени у матичном језику. У упиту, поредак је прецизиран кључном речју ORDER BY иза које следи листа критеријума, од којих сваки може имати спецификатор (ASC за растући поредак, или DESC за опадајући). Поредак одређује релацију $\preceq$ над редовима која је лексикографско проширење релације $\leq$ над низом критеријума[5]. Детерминистички поредак, *дет*-поредак (енгл. *det-ordering*), је или *јединствени*-поредак (енгл. *unique-ordering*), тј. поредак који је одређен вредностима атрибута који имају својство јединствености (на пример, поредак по примарном кључу резултујуће табеле), или пројекциони поредак, *сел*-поредак (енгл. *sel-ordering*),

[5]Случајеви када се захтева опадајући поредак по неком критеријуму (присутан је спецификатор DESC), еквивалентни су и своде се на захтев растућег поретка по том истом критеријуму помноженом са константом -1.

тј. поредак који је одређен вредностима израза који се појављују у SELECT клаузули. Поредак је детерминистички ако је услов $d\mathcal{C}$ задовољен. Листа поретка може укључити више израза него што је неопходно за постизање *дет*-поретка. Треба приметити да је ово карактеристика сваког појединачног упита, па се зато проверава за оба разматрана упита. Ако упит не садржи ORDER BY клаузулу или поредак није детерминистички, онда систем за управљање базом података може сортирати редове резултујуће табеле различито (чак и за исти упит, систем не гарантује да ће редови бити сортирани сваког пута на исти начин). Зато, ако поредак упита није детерминистички, порука упозорења се штампа, али се ипак наставља са анализом сличног упита: упиту се додаје примарни кључ као критеријум поређења ако постоји, или му се додају критеријуми неопходни за постизање *сел*-поретка.

Пример 4.25 Оба поретка у упитима са слике 4.1 јесу детерминистичка, па су одговарајући услови за оба упита испуњени (јер обе листе критеријума поређења садрже примарни кључ производа табела). Зато, у кораку (m4) алгоритма, упозорење није издато. Слика 4.17 садржи услов $d\mathcal{C}$ за леви упит (формула *Инстанца услова детерминистичког поретка*). Додавање новог критеријума, нпр. c_cust_type на крај било које од ове две листе критеријума не би направило никакву разлику.

Услов поретка ($o\mathcal{C}$): Две релације $\preceq'$ и $\preceq''$ дефинисане на основу два поретка јесу еквивалентне ако су компатибилне, тј. ако услов $o\mathcal{C}$ важи. Тада, како су у овој фази анализе већ задовољени *дет*-пореци (јер су или већ присутни у оригиналним упитима, или се разматрају модификовани упити), табеле су сортиране на такав начин да су издвојени изрази над вредностима редова са истом позицијом исти. Ово је моделовано аксиомом *Једнакост критеријума позиционих функција*. Међутим, у случају да је присутан *јединствени*-поредак (што се једноставно може проверити присуством примарног кључа у ORDER BY клаузули), међу-табеле су сортиране на исти начин, и позиционе функције pos' и pos'' су исте. Ово се може закључити из аксиома *Јединственост*, *Кодомен позиционих функција* и *Једнакост критеријума позиционих функција*, али због ефикасности доказивања експлицитно је моделовано аксиомом *Једнакост позиционих функција* (која се користи уместо *Једнакост критеријума позиционих функција* у даљим резоновањима). Ако пореци нису еквивалентни, издаје се упозорење о томе, и наставља се анализа одговарајућих модификованих упита са еквивалентним порецима.

Пример 4.26 У примеру са слике 4.1, пореци су еквивалентни, тј. услов *Инстанца услова поретка* са слике 4.17 је задовољен: $o\mathcal{C}$ обухвата еквивалентност критеријума поређења -t_amount и t_amount DESC, a_avail_blnc + t_amount и a_avail_blnc (јер је већ уређен по t_amount), као и еквивалентност критеријума :in1_sqc - t_txn_date и t_txn_date DESC. Како су ови пореци еквивалентни, и како су *јединствени*-пореци, у склопу корака (m5) у алгоритму формула *Инстанца једнакости позиционих функција* са слике 4.19 додаје се формули која ће се даље користити.

LIMIT клаузула: Присуство LIMIT клаузуле у упиту значајно је због максималног броја итерација које ће бити направљене по курсору у C/C++ језику. Ако су границе различите, и ако се у матичном језику итерира до мање вредности од тих двеју граница, тада та различитост не утиче на еквиваленцију. Међутим, ако се итерира већи број пута, или по свим враћеним вредностима курсора, тада се може десити да еквиваленција није присутна. У циљу поједностављења ове ситуације, разматра се само случај када су

вредности граница једнаке. Како су вредности у `LIMIT` клаузули константе, провера ове једнакости је тривијална и не укључује SMT решавач. Ако вредности нису једнаке, онда, слично као код услова $o\mathcal{C}$, разматраја се еквиваленција два фрагмента кода са истим границама (мањом од иницијалних), и исписује се одговарајућа порука упозорења.

Пример 4.27 За текући пример са слике 4.1, границе су једнаке, па нема никакве поруке у кораку (m6) алгоритма.

Услов пројекција ($s\mathcal{C}$): У случају да су промене анализиране функције направљене само у коду пре упита и у самом упиту, кôд после упита остаје исти, тј. коришћење података добијених из упита остаје исто. Зато се дефинише услов еквиваленције који проверава да ли су вредности враћене упитом једнаке, означен са $s\mathcal{C}$. У случају да је овај услов задовољен и да нема петљи пре упита, пуна еквивалентност је потврђена. Међутим, ако су петље присутне пре упита, онда се разматра k-еквиваленција јер се петље размотавају фиксирани број пута. Тада, m-аксиоме се и не корсите, јер нису неопходне за доказивање услова који се проверава (оне служе да повежу резултате упита са променљивама из C/C++ језика).

Пример 4.28 Како постоје промене и после `SQL` упита у текућем примеру са слике 4.1, услов $s\mathcal{C}$ се не проверава, тј. провере дефинисане у оквиру корака (m7) у алгоритму се не врше.

На крају, у склопу корака (m8) алгоритма, ако је кôд после упита мењан, услов $hl\mathcal{C}$ се проверава.

Пример 4.29 За текући пример са слике 4.1, услов $hl\mathcal{C}$, тј. провера да ли су низови у који се смештају подаци из базе (`days`, `blnc` и др.) једнаки у дужини и садржају, реализује се и потврђује да важи.

У случају да еквиваленција није присутна, одговарајућа порука ће указивати на услов који није задовољен и на тај начин сугерисати програмеру шта треба поправити. На пример, ако *Услов спајања* није испуњен, проблем је у нееквивалентним `FROM` клаузулама, док *Услов филтрирања* указује на проблем у рефакторисаној `WHERE` клаузули. Ако *Услов поретка* није задовољен, проблем је у `ORDER BY` клаузули, а *Услов пројекција* потврђује проблем у `SELECT` клаузулама, тј. у пројектованим вредностима упита.

4.3.3 Проширење приступа: `GROUP BY` и агрегатне функције

Предложени приступ може се проширити да подржи и додатне конструкте SQL језика. Да би се то реализовало, неопходно је проширити:

1. сигнатуру теорије,
2. мета функцију σ,
3. скуп коришћених аксиома,
4. скуп упит-зависних услова еквиваленције и
5. алгоритам за проверу услова еквиваленције.

Илустрација оваквог поступка биће описана кроз додавање подршке за `GROUP BY` клаузулу и агрегатне функције (`avg`, `count`, `max`, `min`, и `sum`). Одговарајуће проширење SQL граматике дато је на слици 4.20.

```
Query ::= SELECT Exprs
          FROM Tables
          [WHERE Cond]
          [GROUP BY Exprs]
          [ORDER BY Crit]
          [LIMIT num]
```

```
Expr ::= ...
       | AVG (Expr)
       | COUNT (Expr)
       | MAX (Expr)
       | MIN (Expr)
       | SUM (Expr)
```

Слика 4.20: Проширење граматике `GROUP BY` клаузулом и агрегатним функцијама.

Сигнатура теорије проширује се скупом $\mathcal{A}_{gg}$ који садржи нове функцијске симболе: avg, sum, $count$, min, max (који ће бити тумачени као неинтерпретиране функције), и новим скупом променљивих $\mathcal{V}_{id}$. Типизирајућа функција α проширује се на следећи начин:

$$\alpha(\phi) \quad = \quad \begin{cases} \ldots & \\ ((tab, num, num), num), & \text{ако } \phi \in \mathcal{A}_{gg} \\ num, & \text{ако } v \in \mathcal{V}_{id} \end{cases}$$

Сваки нови функцијски симбол има придружена три аргумента. Интуитивно, први одговара међу-табели, други (који је променљива из скупа $\mathcal{V}_{id}$) одговара идентификатору упита, док трећи (такође променљива из скупа $\mathcal{V}_{id}$) одговара изразу у агрегатној функцији.

Пример 4.30 Резултат примене функције σ на табелу `T` и израз `min(exp)` је:

$$\sigma\big(r, T, \texttt{min(exp)}\big) = min\big(im(T), id_q, id_{exp}\big)$$

Променљива id_q је фиксирана променљива за разматрани упит. Променљива id_{exp} је различита за синтаксно различите израз који се појављују у агрегатним функцијама.

Пример 4.31 На слици 4.21, за

`min((:in_sqc-a_open_date)/7)` и `max((:in_sqca_open_date)/7)`

одговарајућа променљива је $id_{(:in\_sqc-a\_open\_date)/7}$, док су за

`min((:in_sqc-a_open_date)/7)` и `max((a_last_actvt-a_open_date)/7)`

одговарајуће променљиве $id_{(:in\_sqc-a\_open\_date)/7}$ и $id_{(a\_last\_actvt-a\_open\_date)/7}$.

```
SELECT a_avail_blnc / 1000,
  a_avail_blnc / 1000
  - a_pending_blnc / 1000,
  MAX((a_last_actvt_date-a_open_date)/7)
  MIN((:in_sqc-a_open_date)/7)
FROM account
WHERE a_last_actvt = :in_sqc
GROUP BY a_avail_blnc / 1000,
  a_pending_blnc / 1000
```

```
SELECT a_avail_blnc / 1000,
  a_avail_blnc / 1000
  - a_pending_blnc / 1000,
  MAX((:in_sqc-a_open_date)/7)
  MIN((:in_sqc-a_open_date)/7)
FROM account
WHERE a_last_actvt = :in_sqc
GROUP BY a_avail_blnc / 1000,
  a_avail_blnc / 1000
  - a_pending_blnc / 1000
```

Слика 4.21: Налажење најраније и најскорије отвореног налога на коме је забележена последња активност данас (улазни аргумент упита). Упит враћа време протекло од отварања налога у недељама, груписано по новом и старом стању израженом у хиљадама динара.

Међутим, ради прецизнијег моделовања, за сваки пар $(\texttt{aggexpr}_i, \texttt{aggexpr}_j)$ синтаксно различитих израза који се појављује у агрегатним функцијама, у алгоритам се додаје следећи корак:

$$\texttt{try to prove } \forall \overrightarrow{input}\ (\Phi_{\text{F}} \Rightarrow (\sigma(r, T, \texttt{aggexpr}_i) = \sigma(r, T, \texttt{aggexpr}_j)))$$
$$\texttt{if succeed add } (id_{aggexpr_i} = id_{aggexpr_j}) \texttt{ into } \Phi_{\text{F}}$$

Циљ ове провере је да открије семантички еквивалентне изразе и да им додели једнаке идентификаторе.

Пример 4.32 Ако `WHERE` клаузула упита садржи једнакост `a_last_actvt = :in_sqc`, тада се формули Φ_{F} додаје једнакост

$$id_{(:in\_sqc-a\_open\_date)/7} = id_{(a\_last\_actvt-a\_open\_date)/7}.$$

Пример 4.33 За изразе `x+y` и `y+x` треба додати једнакост $id_{\texttt{x+y}} = id_{\texttt{y+x}}$.

Услов груписања ($g\mathcal{C}$): Скуп аксиома које се користе не мења се у овом случају, док се скуп упит-зависних услова еквиваленције допуњује условом *Услов груписања – $g\mathcal{C}$* (слика 4.22). Интуитивно, овај услов проверава да ли се груписање редова из међутабеле $im(T)$ ради на еквивалентан начин у обе `GROUP BY` клаузуле, тј. да ли се два реда првом `GROUP BY` клаузулом класификују у исту групу ако и само ако се налазе у истој групи одређеној другом `GROUP BY` клаузулом.

Пример 4.34 Пример еквивалентних `GROUP BY` клаузула заједно са агрегатним функцијама дат је на слици 4.21.

$$\forall\, r_1 : row, r_2 : row\ (\sigma(r_1, T', \texttt{con}') \wedge \sigma(r_2, T'', \texttt{con}'') \Rightarrow \qquad (\textit{Услов груписања})$$
$$(\bigwedge_{i=1..n'_{\texttt{gexpr}}} \sigma(r_1, T', \texttt{gexpr}'_i) = \sigma(r_2, T', \texttt{gexpr}'_i) \Leftrightarrow \bigwedge_{i=1..n''_{\texttt{gexpr}}} \sigma(r_1, T'', \texttt{gexpr}''_i) = \sigma(r_2, T'', \texttt{gexpr}''_i))$$

Слика 4.22: Услов груписања ($\texttt{gexpr}_i$ одговара i-том изразу у `GROUP BY` клаузули, док $n_{\texttt{gexpr}}$ одговара њиховом броју).

На крају, алгоритам за проверу услова еквивалентности проширује се додавањем корака приказаних на слици 4.23. Корак (m3.1) проверава да ли услов $g\mathcal{C}$ важи, и ако то није случај закључује да еквивалентност не може бити доказана. У супротном, формули $\Phi_{\bar{\texttt{F}}',\bar{\texttt{F}}''}$ додаје се једнакост $id_{q_1} = id_{q_2}$ која сведочи о еквиваленцији `GROUP BY` клаузула. Корак (m3.2) проверава еквиваленцију сваког пара аргумената агрегатних функција који припадају различитим упитима (сличан корак је већ урађен за сваки упит понаособ), и ако она важи, додаје се у формулу једнакост одговарајућих идентификатора.

Примарни кључ се најчешће не налази у `GROUP BY` клаузулама, јер би у супротном проузроковао свођење сваке групе на један једини ред. Иако ово није очекивано коришћење конструкта `GROUP BY`, ипак се може појавити после неког низа рефакторисања у коду. У таквим случајевима, агрегатне функције у `SELECT` или `ORDER BY` клаузулама би оперисале над једним редом. Предложен приступ може се применити и у овој неуобичајеној ситуацији у циљу доказивања еквивалентности две функције, од којих једна садржи m-упит са, а друга m-упит без `GROUP BY` клаузуле.

(m3.1) try to prove $\forall\, \overrightarrow{input}\ (\Phi_{\bar{\texttt{F}}',\bar{\texttt{F}}''} \Rightarrow g\mathcal{C})$
if failed
print „Group by clauses are not equivalent!"
return cannot prove equivalence
else add $(id_{q_1} = id_{q_2})$ into $\Phi_{\texttt{F}',\texttt{F}''}$, $\Phi_{\bar{\texttt{F}}',\bar{\texttt{F}}''}$

(m3.2) for each $\texttt{aggexpr}_i$ from $\Phi'_{\texttt{F}}$ and for each $\texttt{aggexpr}_j$ from $\Phi''_{\texttt{F}}$
try to prove $\forall\, \overrightarrow{input}\ (\Phi_{\bar{\texttt{F}}',\bar{\texttt{F}}''} \Rightarrow \sigma(r, T', \texttt{aggexpr}_i) = \sigma(r, T'', \texttt{aggexpr}_j))$
if succeed add $(id_{\texttt{aggexpr}_i} = id_{\texttt{aggexpr}_j})$ into $\Phi_{\texttt{F}',\texttt{F}''}$, $\Phi_{\bar{\texttt{F}}',\bar{\texttt{F}}''}$

Слика 4.23: Проширење алгоритма са слике 4.16 додавањем корака (m3.1) и (m3.2) између постојећих корака (m3) и (m4).

4.3.4 Логички статус моделовања

Нека је Ψ својство које важи за свако конкретно извршавање функција F' и F'' позваних над истим улазима, што ће бити означено са $\text{F}', \text{F}'' \models \Psi$. *Сагласност* предложеног приступа значи да $\Phi_{\text{F}',\text{F}''} \vdash \Psi$ повлачи $\text{F}', \text{F}'' \models \Psi$, док *потпуност* значи обратно.

У неколико аспеката предложени приступ се природно изражава у оквиру логике вишег реда пре него у терминима логике првог реда. Међутим, покушано је да се сва аутоматска резоновања врше у оквиру FOL-а, како би се користили ефикасни доказивачи и SMT решавачи. Такође, предложено моделовање није прецизно у неким аспектима. На пример, разматрају се само еквивалентна филтрирања, иако је могуће да се у матичном језику уради додатна филтрирања која ће очувати укупну еквивалентност. Из тих разлога, предложени приступ није потпун, али и даље садржи релевантне информације за доказивање великог броја битних случајева еквиваленције. Финији детаљи у моделовању су могући, и водили би ка различитим нивоима изражајности.

Поверење у сагласност приступа лежи у релативно простом и интуитивно прихватљивом скупу коришћених аксиома, и такође у чињеници да су сви негативни тест примери (где две разматране функције нису међусобно еквивалентне) из корпуса (видети поглавље 4.4) откривени, а за позитивне примере је доказана еквивалентност. Сагласност присупа у коју бисмо били потпуно сигурни би могла да се докаже коришћењем неког асистента при доказивању (енгл. *proof assistant*), али би то додатно захтевало огроман труд и дуго време. Такође, сагласност и потпуност приступа зависе од коришћене теорије и прецизности којом она описује релевантна израчунавања.

4.4 Имплементација и експериментална евалуација

Предложени приступ је имплементиран у виду алата SQLAV, који је јавно доступан и отвореног је кода [196]. У овом поглављу, представљени су његови имплементациони детаљи, корпус примера за тестирање и евалуацију алата на њима.

4.4.1 Имплементациони детаљи

Алат SQLAV је имплементиран је у језику *python* и користи алате LAV и SQL2FOL. У потпуности прати архитеткуру приказану на слици 4.3 и 4.12, као и алгоритам са слике 4.16. Подржава упите који су описани граматиком датом у поглављу 4.2.1 и проширеној у поглављу 4.3.3.

Алат SQL2FOL је написан у језику C++ и користи алате *Lex* и *Bison* за лексичку и синтаксну анализу, редом. На основу општих образаца аксиома, генерише аксиоме које одговарају конкретним уграђеним SQL упитима, као и упит-зависне услове еквивалентности, док LAV генерише описе функција у матичном језику и услове еквивалентности.

Све формуле које ови алати генеришу су у SMT-lib формату, и обрађују се као спољне датотеке, па се могу користити различити решавачи/доказивачи. У експерименталној евалуацији чији су резултати дати у поглављу 4.4.4, коришћен је SMT решавач *Z3* [67] и FOL доказивач *Vampire* [158]. Алат је модуларан и дозвољава интеграцију додатног изражајнијег моделовања.

Развојни процес кроз *GitHub* акције

Са идејом да се предложени алат учини лако доступним у свакодневној употреби, имплементиран је процес континуалне интеграције (енгл. *continuous integration workflow*) који се директно може додати у било који *GitHub* репозиторијум користећи *GitHub* акције[6] (енгл. *GitHub Actions*).

GitHub акције дозвољавају покретање процеса на основу неког одређеног *GitHub* догађаја, нпр. када неко унесе своје измене у репозиторијум (енгл. *pushing a commit*), захтева довлачење нове верзије (енгл. *pull request*), и слично. Ова пракса у развоју софтвера најчешће укључује назубљивање кода, анализу кода у смислу тражења стилских грешака и сумњивих конструката (енгл. *code linters*), безбедносне и функционалне провере. Такође је погодна и за било какве друге специфичне провере, као у овом случају, где су имплементирана два различита процеса провера. Прецизније, развијена су два различита начина на који се може контролисати функционална еквивалентност:

- провера еквивалентности на захтев (енгл. *on-demand*) и
- провера еквивалентности после сваког уношења измена у репозиторијум.

Процес на захтев је користан када програмер жели да провери коректност својих тренутних промена, без уношења измена у репозиторијум. У оваквој ситуацији, програмер може ручно иницирати процес *GitHub* акције који садржи алат SQLAV користећи *GitHub* API и догађај обавештења репозиторијума (енгл. *repository dispatch event*) помоћу POST захтева (енгл. *post request*) са ауторизацијом за приступ репозиторијуму [75]. На овај начин, програмер добија потврду да ли начињене промене чувају функционалну еквивалентност са претходном верзијом. У случају да добије поруку која указује где је нарушена еквиваленција, може решити проблем пре уношења измена кода у репозиторијум. Описани поступак може уштедети време програмеру и спречити уношење измењеног кода који не чува еквивалентност и који може увести нове грешке и проблеме у репозиторијум. Други процес је користан у ситуацијама када се захтева обавезна провера еквивалентности по уношењу промена, која се врши аутоматски.

Описани процеси су дати у виду јавно доступних конфигурационих YAML фајлова [177], као и скриптова који се користе у процесима са циљем припреме улазног фајла за алат SQLAV на основу тренутних промена (јер SQLAV захтева две функције чија се еквивалентност проверава, док је у овим процесима само рефакторисана верзија кода доступна, а оригинална се мора повући из репозиторијума).

Сви тест примери из корпуса представљеног у следећем поглављу (осим оних за испитивање скалабилности) додати су у *GitHub* репозиторијум [177] у форми две верзије, оригиналне и измењене. Први садржи иницијалну верзију функције, док други садржи њену рефакторисану верзију за коју треба аутоматски проверити еквивалентност покретањем процеса који користи SQLAV алат. Одговарајућа страница са *GitHub* акцијама садржи зелене знаке штиклирања за све примере где је еквивалентност доказана (позитивни примери) и црвене X-знаке у супротном (негативни примери), заједно са пропратном поруком која указује на место где је нарушена еквивалентност.

[6] `https://github.com/features/actions`

4.4.2 Скупови примера за тестирање коришћени у евалуацији

У тренутку писања ове тезе, не постоји доступан корпус који се може користити у сврхе евалуације система који разматра обостране промене у уграђеним упитима и матичном језику. У циљу анализе и евалуације предложеног решења, овај проблем је превазиђен конструкцијом релевантног корпуса ове сврхе. Осим тога, корпус може бити и јавно доступна полазна основа за поређење различитих алата ове врсте у будућности. Дизајниран је са идејом да покрије велики број случајева коришћења подржаних у тренутној верзији предложеног система.

Анализирани су примери из више књига и уџбеника из области уграђеног SQL-а и SQL туторијала. Код првих, обично су дате кратке смернице како уградити SQL упите у језик опште намене, и претпоставља се да је читалац већ искусан са базама података и SQL језиком, па је у њима јако мало примера и детаља. Са друге стране, SQL туторијали покривају упите врло детаљно, али не садрже интеракцију са императивним језиком. Узимајући у обзир предности и мане оба скупа литературе, сакупљен је корпус на основу упита из често препоручиване књиге у области база података и SQL-а [19].

Изабрана су 33 потпуно различита упита и уграђени су у C/C++ функције, сваки упит на два начина, добијајући две различите али еквивалентне имплементације. Такође, додате су и три функције са различитим бројем уграђених *s*-упита, који користе схему базе података из изабране књиге. У циљу испитивања скалабилности, користе се различите вредности параметра k за проверу k-еквивалентности у 14 примера са курсорима, али су такође додати нови примери мењањем броја пројекција у свим *m*-упитима тако да имају од једне до пет пројекција за сваки *m*-упит (како их има 26, на овај начин добијено је $4 \cdot 26$ додатних примера). Све укупно, корпус садржи 140 примера.

Анализом структуре упита у изграђеном копусу, добијено је да има 28% уграђених *s*-упита (17% *su*-упита и 11% *st*-упита), и 72% уграђених *m*-упита (11% *mu*-упита и 61% *mt*-упита). У 46% примера са *mu*-упитима, измене су направљене у C/C++ језику пре упита и у самом упиту, у 35% случајева промене су у упиту и после њега, док се у остатку (19%) промене протежу кроз сва три дела кода. У 14% примера користе се погледи, и истом проценту су заступљена и груписања, док је агрегација присутна у 17% корпуса.

Такође, незнатним променама свих већ описаних примера, додато је нових 140 негативних примера (што дуплира корпус на укупно 280 тест примера), тј. оних који представљају нееквивалентне парове функција. Ове измене представљају типичне програмерске грешке у императивном језику (нпр. погрешан индекс, оператор $<$ коришћен уместо $\leq$, неинкрементирање бројача, тотално заборављену доделу или return наредбу, и др.), и грешке у SQL језику (приступ погрешном атрибуту, заборављање кључне речи `DESC`, погрешне агрегатне функције, пропуштен израз у `SELECT`, `ORDER BY` или `GROUP BY` клаузули, грешке у куцању имена табеле или атрибута, и др.). Неки негативни тест примери направљени су тако што није дефинисан примарни или страни кључ.

4.4.3 Експериментално окружење

За потребе евалуације алата SQLAV, коришћено је идентично окружење као у поглављу 3.8, тј. сви експерименти представљени у наставку покретани су на систему са процесором ознаке *Intel® Core™ i7-4790 CPU @ 3.60GHz* са осам језгара, 16GB RAM меморије, и оперативним системом *Ubuntu 18.04.3 LTS*.

Заједно са представљеним корпусом, доступан је и алат SQLAV (као и сви резултати мерења приказани у наставку) [196], па се сви експерименти могу у потпуности поновити.

4.4.4 Резултати евалуације

Резултати евалуације алата SQLAV сумирани су у табелама 4.1 и 4.2. Средње вредности времена потребних за изградњу и проверу услова у формулама израженим у терминима теорије $\mathbf{L}^a$ дате су у секундама. Резултати за теорију $\mathbf{B}^a$ нису приказани јер *Vampire* не подршава бит-векторе, а времена за решавач *Z3* су између 2 и 100 пута дужа него код теорије $\mathbf{L}^a$. Постављено ограничење у трајању (енгл. *timeout*) било је 60 секунди. Предложени систем генерише потребне формуле врло ефикасно, за мање од 0.3 секунде у већини примера.

Табела 4.1: Средње вредности времена доказивања у секундама за: генерисање формула са условима, решавање сваког услова појединачно, и укупно време верификације; број нерешених примера због временског ограничења, ако постоје, дат у заградама; одговарајући интервали за вредности $median/mean$, означени са md/m, дати за сваку групу резултата

Тип упита		Еквивалентни			Нееквивалентни	
		Број парова			Број парова	
s-упити	ген.	10	0.061		10	0.059
			Z3	Vampire		Z3
	$hl\mathcal{C}$	10	0.017 (3)	0.121 (1)	10	0.018 (3)
	укупно	10	0.078 (3)	0.182 (1)	10	0.076 (3)
	md/m		[1.01 - 1.03]	[0.67 - 0.75]		[1.00 - 1.03]
m-упити	ген.	12	0.151		12	0.138
			Z3	Vampire		Z3
(пуна ек.)	$j\mathcal{C}$	12	0.001	0.001	12	0.001
	$f\mathcal{C}$	12	0.019	0.055	11	0.019
	$d\mathcal{C}$	24	0.014	0.042	8	0.019
	$o\mathcal{C}$	12	0.011	0.028	4	0.010
	$g\mathcal{C}$	2	0.011	-	1	0.019
	$s\mathcal{C}$	12	0.012	0.026	4	0.019
	укупно	12	0.223	0.335	12	0.189
	md/m		[0.77 - 1.07]	[0.71 - 1.01]		[1.00 - 1.09]
m-упити	ген.	14	0.160		14	0.173
			Z3	Vampire		Z3
(к=5)	$j\mathcal{C}$	14	0.001	0.001	14	0.001
	$f\mathcal{C}$	14	0.018	0.053 (4)	13	0.018
	$d\mathcal{C}$	28	0.015	0.075 (3)	20	0.016
	$o\mathcal{C}$	14	0.011	0.028	10	0.011
	$g\mathcal{C}$	2	0.020	-	2	0.022
	$hl\mathcal{C}$	14	0.721	- (7)	10	2.097
	укупно	14	0.973	- (14)	14	1.748
	md/m		[0.65 - 0.99]	[0.37 - 1.00]		[0.85 - 1.14]

S-упити. Z3 није успео да реши услед временског ограничења три примера уграђених *s*-упита из корпуса (у сваком од њих се користи аксиома *Јединствености*), док је код доказивача *Vampire* само један такав.

Остали примери се доказују/оповргавају врло ефикасно. Код нееквивалентних примера, *Vampire* није успео да реши ни један, па те информације нису присутне у табели.

M-упити. У паровима примера који се разликују у коду пре упита и у самом упиту, доказује се пуна еквивалентност (услов $s\mathcal{C}$ и остали потребни услови), док се за остале примере разматра k-еквиваленција, за вредности $k = 5$ (услов $hl\mathcal{C}$ и остали потребни услови). Просечно време потребно за доказивање пуне еквиваленције је 0.335 секунди када се користи *Vampire*, и 0.223 секунди када се користи *Z3*, и већина овог времена одлази на генерисање формула ($45 - 68\%$). Просечно време потребно за доказивање k-еквиваленције је знатно дуже, 0.973 секунди при коришћењу решавача *Z3*, и у овом случају доминира време потребно за доказивање $hl\mathcal{C}$ услова (74%), док ове сложеније услове *Vampire* и не успева да докаже у задатом временском интервалу. У случају негативних тест примера, укупно време јако зависи од места на коме је пронађена разлика. *Vampire* опет није успео да реши ни један пример, али је *Z3* био врло ефикасан без иједног прекорачења временског ограничења.

Табела 4.2: Број пројекција/итерација петље и одговарајуће средње вредности времена потребног за верификацнију у секундама; број нерешених примера због временског ограничења, ако постоје, дат у заградама; одговарајући интервали за вредности $median/mean$, изначени са md/m, дати за сваку групу резултата

		Еквивалентни		Нееквивалентни
m-упит	Број парова	Z3	Vampire	Z3
број пројекција				
1	26	0.365	0.254 (14)	0.679
2	26	0.526	0.312 (17)	0.787
3	26	0.674	0.361 (15)	0.960
4	26	0.802	0.406 (15)	1.085
5	26	0.949	0.448 (15)	0.986
md/m		[0.53 - 0.91]	[0.93 - 1.03]	[0.25 - 0.31]
број итерација				
(к = 1)	14	0.183	- (14)	0.205
(к = 2)	14	0.214	- (14)	0.235
(к = 3)	14	0.311	- (14)	0.397
(к = 4)	14	0.526	- (14)	0.817
(к = 5)	14	0.976	- (14)	1.734
(к = 6)	14	1.870	- (14)	2.366
(к = 7)	14	4.203	- (14)	5.248
(к = 8)	14	9.538	- (14)	5.642
(к = 9)	14	21.255	- (14)	17.540
(к = 10)	14	39.799 (3)	- (14)	12.500 (3)
(к = 11)	14	32.303 (12)	- (14)	15.845 (4)
(к = 12)	14	37.292 (13)	- (14)	17.432 (5)
(к = 13)	14	55.775 (13)	- (14)	26.056 (5)
md/m		[0.84 - 1.08]	-	[0.33 - 1.21]

Разматрајући скалабилност, експериментални резултати показују да повећавање вредности k не утиче на проверу упит-зависних услова, али знатно усложава проверу $hl\mathcal{C}$ услова. Како је и очекивано, потребно време расте експоненцијално, што показује да се овај приступ не може користити ефикасно за веће вредности параметра k. Са друге стране, повећање броја пројекција, за све примере, утиче на ефикасност само линеарно.

Главни закључак ове евалуације сугерише да је представљени приступ врло ефикасан у већини случајева, због чега је применљив у реалним ситуацијама и могао би се користити у интегрисаним развојним окружењима као алат који би свакодневно помагао програмерима.

Глава 5

Закључци и даљи рад

У овом поглављу дати су главни закључци дисертације, како теоријски тако и практични, као и могући правци даљег истраживања који би се наслањали на резултате приказане у овој тези.

5.1 Главни доприноси тезе

У овој докторској дисертацији развијена су и представљена моделовања упитних језика SPARQL и SQL у логици првог реда. Она су омогућила аутоматско резоновање о проблемима садржаности, еквивалентности и задовољивости упита у језику SPARQL, као и разматрање еквивалентности императивног кода са уграђеним SQL упитима, што и демонстрира оригинални приступ приказан у тези. Проблеми се своде на проблеме задовољивости формуле у теоријама логкике првог реда и на тај начин омогућава се коришћење ефикасних SMT решавача.

У случају језика SPARQL, приступ покрива најчешће коришћене језичке конструкте (конјунктивне и цикличне упите, клаузуле `filter`, `union`, `optional`, `from`, неименоване чворове, пројекције и подупите). Доказана је сагласност и потпуност описаног приступа у складу са семантиком програмског језика, тј. ако процедура одлучивања приказана у тези за два упита каже да су у релацији садржаности или еквивалентности (или да је упит незадовољив), тада ти упити заиста задовољавају ту релацију (или упит заиста јесте незадовољив), и супротно, ако упит(и) задовољава(ју) једну од ове три релације, процедура ће то и потврдити.

Демонстрирана је проширивост представљених дефиниција и доказа тврђења у случају додавања подршке за нове типове граф образаца, израза и услова (конструкти `diff` и `graph` и уграђене функције), при чему је сагласност (и потпуност у неким случајевима) одржана. Такође, представљено је и резоновање о проблему садржаности узимајући у обзир RDF схему, која намеће ограничења на интерпретацију графова над којима се упити извршавају и омогућава имплицитно проширење података у графу додавањем нових триплета који се могу закључити из правила RDF схеме.

Проблем садржаности разматран је и у својој стандардној форми, и као релација стапања. У пракси, разматрање ове релације је врло уобичајено, узимајући у обзир непотпуне резултате SPARQL упита који се могу појавити. Заменом релације садржаности њеном слабијом формом стапања, дефиниције и тврђења теореми проширују се врло природно и интуитивно.

Предложени приступ је имплементиран у оквиру алата отвореног кода, под називом SpeCS, чије су перформансе показане на два различита корпуса тестова: три групе из скупа тестова *Query Containment Benchmark*, и десет који су генерисани алатом *SQC-Framework*. Представљена евалуација показује огромне предности решавача SpeCS у поређењу са другим савременим решавачима, како са аспекта ефикасности, тако и у смислу покривености језичких конструката. Откривени су и отклоњени примећени недостаци у постојећим скуповима тестова. У оквиру имплементације, подржано је и откривање релације садржаности и у случају преименовања пројектованих променљивих, које је као могућност важно и корисно у пракси, па је самим тим присутно и у примерима тестова из корпуса генерисаног алатом *SQCFramework*.

Такође, представљена је једна примена моделовања упитних језика са циљем ефикасног аутоматског резоновања о еквивалентности функција са уграђеним SQL упитима. Еквиваленција се проверава у контексту рефакторисања и промена које укључују и SQL и императивни кôд. За конструисање услова еквиваленције, развијена је у логици првог реда посебна семантика заснована на аксиомама. У зависности од упита и од контекста у ком се појављује, конструишу се услови еквивалентности. За резовновање над тим условима такође се могу користити SMT решавачи, као и доказивачи у логици првог реда.

Представљени приступ дозвољава померање рачунања из матичног језика у SQL и обратно. Такође дозвољава и уклањање непотребних филтера и критеријума сортирања у исто време, а може открити и да нека спајања табела нису потребна. Предложени приступ има модуларну архитектуру која дозвољава испитивање еквивалентности део по део, а не само глобално — доказивање еквивалентности може доживети неуспех при провери било ког услова. Ово позитивно утиче како на ефикасност, тако и на скалабилност приступа. Такође, у случајевима када се еквиваленција оповргава, грануларни приступ помаже у разумевању проблема тако што пружа информацију који од услова еквиваленције није задовољен. У супротном, приступ у поређењу са тестирањем пружа већи степен поузданости у рефакторисан кôд. На пример, доказана k-еквивалентност за кôд који има само једну улазну променљиву већ за вредности $k = 2$ замењује милијарде различитих случајева[1] за тестирање. Додатно, ово гарантује потпуну покривеност кода који има у себи петље које се размотавају највише k пута. На крају, модуларна архитектура дозвољава проширивање приступа додавањем подршке за додатне SQL конструкте.

Приступ је имплементиран кроз алат SQLAV чије су перформансе показане на корпусу примера са уграђеним SQL-ом у програмски језик C/C++. SMT решавач *Z3* и FOL доказивач *Vampire* су коришћени за проверу ваљаности генерисаних формула. У недостатку одговарајућег корпуса за евалуацију, конструисан је скуп од више стотина примера који представља добру полазну основу за поређење различитих алата ове врсте у будућности. Резултати евалуације на њему показују да је предложени приступ ефикасан и да има потенцијала да буде применљив као подршка процесу рефакторисања. Да би се олакшало његово свакодневно коришћење, алат је интегрисан у два процеса *GitHub* акција: један који се може користити за проверу еквивалентности на захтев и други за аутоматску проверу еквивалентности после сваког уношења измена у репозиторијум.

[1] Велика већина ових случајева је еквивалентна.

5.2 Могућа унапређења

Више је могућности за даље правце истраживања на тему садржаности SPARQL упита. Један аспект подразумева даље проширивање језичке покривености приступа, нпр. подржавајући нове граф обрасце, друге типове услова, израза и уграђених функција, као и мање заступљене конструкте. У случају таквих проширења, врло је битно задржати сагласност и потпуност приступа (ако је могуће), и тако потврдити ту могућност представљену у тези.

При резоновању о релацији садржаности, може се разматрати шири скуп аксиома. Један такав пример су $\mathcal{SHI}$ аксиоме, које представљају један фрагмент дескриптивне логике који моделује релевантни домен у погледу појмова концепата и улога (енгл. *concepts and roles*), и већ постоје истраживања која и њих узимају у обзир.

У плану је примена сличног приступа на друге графовске упитне језике (нпр. XPath, GQL), сродне језику SPARQL. Такође, имајући у виду све више доступних података у RDF формату и апликација из области Семантичког веба које им приступају користећи SPARQL упите, вреди размотрити могућности примене представљеног моделовања у контексту рефакторисања кода са овим типом упита, на сличан начин као што је то урађено за језик SQL уграђен у C/C++ кôд.

Постоји неколико могућих смерова за даљи рад на моделовању уграђеног SQL-а. Један од њих је проширење подржаног подскупа SQL-а. Ово је главни правац даљих унапређења приступа који не претендује да замени постојеће напредне приступе у рефакторисању SQL-а, већ треба да се користи као њихова допуна која покрива овакву врсту рефакторисања.

Када се еквиваленција две функције са уграђеним SQL-ом оповргне, поред прецизирања услова еквивалентности који није задовољен, било би корисно да корисник добије и контра-пример. У плану је истраживање могућности генерисања контра-примера.

Још један правац унапређења представљеног приступа јесте подржавање различитих типова података. У случају да колоне имају као своје вредности ниске, може се користити упрошћени модел који их третира као позиве неинтерпретираних функција. Очекује се да нека занимљива својства уграђених функција могу бити моделована на овај начин, док би за моделовање подниски и надовезивања ниски било неопходно користити прецизније моделовање (рецимо коришћењем теорије низова или неке друге теорије).

Још једна могућност за унапређење је ослобађање од претпоставке да постоји само један m-упит по анализираној функцији. Иначе, то не представља велико ограничење имајући у виду да кôд треба рефакторисати у малим корацима, од којих је сваки пропраћен провером еквивалентности. Међутим, овај недостатак условљава да приступ подржава само погледе без `ORDER BY` и `LIMIT` клаузули. У супротном, погледима који их садрже одговарале би ситуације са више од једног m-упита по функцији.

У овом тренутку, еквиваленција се не може доказати ако је филтрирање у једној функцији урађено у SQL упиту, док је у другој реализовано ван, тј. у матичном језику. Као пример могућег унапређења алата планирано је даље истраживање овог случаја ослабљивањем услова *Услов филтрирања*, тј. провером релације подскуп/надскуп између филтера, или налажењем заједничког подскупа филтера ова два упита.

Аспект који такође није разматран је могућност постојања разлика у прецизности одговарајућих нумеричких типова података између SQL и C/C++ језика. Стандард језика C/C++ чак и не прецизира величину ових података. Код прецизне анализе

ове разлике треба узети у обзир, укључујући разматрање целобројног прекорачења и поткорачења.

Сви ови правци даљег истраживања могу довести до интересантних и корисних научних резултата, како теоријских, тако и практичних у виду новоразвијених алата или надоградње постојећих и њиховог прилагођавања новим контекстима.

Литература

[1] A. Aho, Y. Sagiv, and J. Ullman. “Equivalences Among Relational Expressions.” In: SIAM J. Comput. 8 (1979), pp. 218–246.

[2] Altibase. Altibase Enterprise Grade Open Source Database. on-line at: http://altibase.com/, retrieved November 1st, 2020. 2020.

[3] T. Amtoft, S. Bandhakavi, and A. Banerjee. “A Logic for Information Flow in Object-oriented Programs.” In: POPL. Charleston, South Carolina, USA: ACM, 2006, pp. 91–102. DOI: 10.1145/1111037.1111046.

[4] R. Angles, J. B. Antal, A. Averbuch, P. Boncz, O. Erling, A. Gubichev, V. Haprian, M. Kaufmann, J. L. L. Pey, N. Martínez, J. Marton, M. Paradies, M.-D. Pham, A. Prat-Pérez, M. Spasić, B. A. Steer, G. Szárnyas, and J. Waudby. The LDBC Social Network Benchmark. 2020. arXiv: 2001.02299 [cs.DB].

[5] R. Angles and C. Gutierrez. “The Expressive Power of SPARQL.” In: Proceedings of the 7th International Conference on The Semantic Web. ISWC ’08. Karlsruhe, Germany: Springer, 2008, pp. 114–129. ISBN: 978-3-540-88563-4. DOI: 10.1007/978-3-540-88564-1_8. URL: http://dx.doi.org/10.1007/978-3-540-88564-1_8.

[6] R. Angles and C. Gutierrez. “The Multiset Semantics of SPARQL Patterns.” In: The Semantic Web – ISWC 2016. Ed. by P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil. Springer, 2016, pp. 20–36. ISBN: 978-3-319-46523-4.

[7] R. Angles and C. Gutiérrez. “Negation in SPARQL.” In: Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, May 8-10, 2016. Ed. by R. Pichler and A. S. da Silva. Vol. 1644. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1644/paper11.pdf.

[8] R. Angles and C. Gutiérrez. “Subqueries in SPARQL.” In: Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9-12, 2011. Ed. by P. Barceló and V. Tannen. Vol. 749. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-749/paper19.pdf.

[9] M. Arenas and J. Perez. “Querying semantic web data with SPARQL.” In: Jan. 2011, pp. 305–316. DOI: 10.1145/1989284.1989312.

[10] S. Auer, J. Lehmann, and A.-C. N. Ngomo. “Introduction to Linked Data and Its Lifecycle on the Web.” In: Reasoning Web. Semantic Technologies for the Web of Data. Springer, 2011, pp. 1–75. DOI: 10.1007/978-3-642-23032-5_1.

[11] J. S. Auerbach, M. Hirzel, L. Mandel, A. Shinnar, and J. Siméon. “Handling Env. in a Nested Relational Algebra with Combinators and an Implementation in a Verified Query Compiler.” In: SIGMOD. ACM, 2017, pp. 1555–1569. ISBN: 978-1-4503-4197-4. DOI: 10.1145/3035918.3035961.

[12] D. Babic and A. J. Hu. “Calysto: Scalable and Precise Extended Static Checking.” In: ICSE. ACM, 2008, pp. 211–220. DOI: 10.1145/1368088.1368118.

[13] J. Backes, S. Person, N. Rungta, and O. Tkachuk. “Regression Verification Using Impact Summaries.” In: SPIN. 2013, pp. 99–116. DOI: 10.1007/978-3-642-39176-7_7.

[14] M. Barnett and R. Leino. “Weakest-Precondition of Unstructured Programs.” In: Proceedings of the 6th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. PASTE ’05. New York, NY, USA: ACM, 2005, pp. 82–87. ISBN: 1595932399. DOI: 10.1145/1108792.1108813.

[15] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. “Satisfiability Modulo Theories.” In: Handbook of Satisfiability. Vol. 185. IOS Press, 2009. Chap. 26, pp. 825–885.

[16] C. Barrett, A. Stump, and C. Tinelli. The SMT-LIB Standard: Version 2.0. Tech. rep. on-line at: www.SMT-LIB.org, retrieved November 1st, 2020. Department of Computer Science, The University of Iowa, 2010.

[17] C. Barrett and C. Tinelli. “Satisfiability modulo theories.” In: Handbook of Model Checking. Springer, 2018, pp. 305–343.

[18] G. Barthe, B. Grégoire, C. Kunz, Y. Lakhnech, and S. Z. Béguelin. “Automation in Computer-Aided Cryptography: Proofs, Attacks and Designs.” In: CPP. 2012, pp. 7–8. DOI: 10.1007/978-3-642-35308-6_3.

[19] A. Beaulieu. Learning SQL. 2nd. O’Reilly Media, Inc., 2009. ISBN: 9780596520830.

[20] I. Ben-Gan. Microsoft SQL Server 2012 T-SQL Fundamentals. Pearson Education.

[21] M. Benedikt, G. Konstantinidis, G. Mecca, B. Motik, P. Papotti, D. Santoro, and E. Tsamoura. “Benchmarking the Chase.” In: PODS. ACM, 2017, pp. 37–52. ISBN: 978-1-4503-4198-1. DOI: 10.1145/3034786.3034796.

[22] V. Benzaken, É. Contejean, C. Keller, and E. Martins. “A Coq Formalisation of SQL’s Execution Engines.” In: Interactive Theorem Proving. Ed. by J. Avigad and A. Mahboubi. Springer, 2018, pp. 88–107. ISBN: 978-3-319-94821-8. DOI: https://doi.org/10.1007/978-3-319-94821-8_6.

[23] V. Benzaken and É. Contejean. “A Coq Mechanised Formal Semantics for Realistic SQL Queries: Formally Reconciling SQL and Bag Relational Algebra.” In: Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs. Cascais, Portugal: ACM, 2019, pp. 249–261. ISBN: 978-1-4503-6222-1. DOI: 10.1145/3293880.3294107.

[24] Y. Bertot and P. Castran. Interactive Theorem Proving and Program Development: Coq’Art The Calculus of Inductive Constructions. Springer, 2010. ISBN: 9783642058806. DOI: 10.1007/978-3-662-07964-5.

[25] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. “A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World.” In: Commun. ACM 53.2 (Feb. 2010), pp. 66–75. ISSN: 0001-0782. DOI: 10.1145/1646353.1646374.

[26] K. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh, C.-C. Kanne, F. Özcan, and E. Shekita. “Jaql: A Scripting Language for Large Scale Semistructured Data Analysis.” In: PVLDB 4 (Aug. 2011), pp. 1272–1283. DOI: 10.14778/3402755.3402761.

[27] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. “Design and Implementation of a Special-Purpose Static Program Analyzer for Safety-Critical Real-Time Embedded Software.” In: The Essence of Computation: Complexity, Analysis, Transformation. Ed. by T. Æ. Mogensen, D. A. Schmidt, and I. H. Sudborough. Springer, 2002, pp. 85–108. ISBN: 978-3-540-36377-4. DOI: 10.1007/3-540-36377-7_5.

[28] M. Blome, C. Robertson, S. Cai, M. Jones, M. Sebolt, and G. Hogenson. Open Database Connectivity (ODBC), Microsoft Docs. on-line at: https://docs.microsoft.com/en-us/cpp/data/odbc/open-database-connectivity-odbc?view=vs-2019, retrieved November 1st, 2020.

[29] R. F. Boyce, D. D. Chamberlin, W. F. King, and M. M. Hammer. “Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage.” In: Commun. ACM 18.11 (Nov. 1975), pp. 621–628. ISSN: 0001-0782. DOI: 10.1145/361219.361221.

[30] B. Brumm. Beginning Oracle SQL for Oracle Database 18c. Berkeley, CA: Apress, 2019. ISBN: 978-1-4842-4430-2.

[31] C. Cadar, D. Dunbar, and D. Engler. “KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs.” In: OSDI’08. San Diego, California: USENIX Association, 2008, pp. 209–224.

[32] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. “Ontop: Answering SPARQL queries over relational databases.” In: Semantic Web 8 (Feb. 2016). DOI: 10.3233/SW-160217.

[33] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. “Description Logic Framework for Information Integration.” In: Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 1998). 1998, pp. 2–13.

[34] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. “Containment of Conjunctive Regular Path Queries with Inverse.” In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning. KR’00. Breckenridge, Colorado, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 176–185.

[35] S. Ceri and G. Gottlob. “Translating SQL Into Relational Algebra: Optimization, Semantics, and Equivalence of SQL Queries.” In: IEEE Trans. Softw. Eng. 11.4 (Apr. 1985), pp. 324–345. ISSN: 0098-5589. DOI: 10.1109/TSE.1985.232223.

[36] D. D. Chamberlin and R. F. Boyce. “SEQUEL: A Structured English Query Language.” In: Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control. SIGFIDET ’74. Ann Arbor, Michigan: Association for Computing Machinery, 1974, pp. 249–264. ISBN: 9781450374156. DOI: 10.1145/800296.811515.

[37] E. P. F. Chan. “Containment and Minimization of Positive Conjunctive Queries in OODB’s.” In: Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS ’92. San Diego, California, USA: ACM, 1992, pp. 202–211. ISBN: 0897915194. DOI: 10.1145/137097.137869. URL: https://doi.org/10.1145/137097.137869.

[38] A. K. Chandra and P. M. Merlin. “Optimal Implementation of Conjunctive Queries in Relational Data Bases.” In: STOC. Boulder, Colorado, USA: ACM, 1977, pp. 77–90. DOI: 10.1145/800105.803397.

[39] C. C. Chang and H. J. Keisler. “Model Theory.” In: Journal of Symbolic Logic 41.3 (1976), pp. 697–699. DOI: 10.2307/2272047.

[40] S. Chaudhuri and M. Y. Vardi. “Optimization of Real Conjunctive Queries.” In: PODS. ACM, 1993, pp. 59–70. ISBN: 0-89791-593-3. DOI: 10.1145/153850.153856.

[41] M. Chekol and J. Euzenat. SPARQL Query Containment Benchmark. on-line at: http://sparql-qc-bench.inrialpes.fr/, retrieved November 1st, 2020.

[42] M. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. “SPARQL Query Containment under Schema.” In: Journal on Data Semantics 7.3 (Apr. 2018), pp. 133–154. DOI: 10.1007/s13740-018-0087-1.

[43] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. “SPARQL Query Containment under RDFS Entailment Regime.” In: Automated Reasoning. Ed. by B. Gramlich, D. Miller, and U. Sattler. Springer, 2012, pp. 134–148. ISBN: 978-3-642-31365-3.

[44] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. “SPARQL Query Containment Under SHI Axioms.” In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. AAAI’12. Toronto, Ontario, Canada: AAAI Press, 2012, pp. 10–16. URL: http://dl.acm.org/citation.cfm?id=2900728.2900730.

[45] C. Chekuri and A. Rajaraman. “Conjunctive query containment revisited.” In: Database Theory — ICDT ’97. Ed. by F. Afrati and P. Kolaitis. Springer, 1997, pp. 56–70. ISBN: 978-3-540-49682-3.

[46] R. Chirkova. “Combined-semantics Equivalence of Conjunctive Queries: Decidability and tractability results.” In: J. Comput. Syst. Sci. 82.3 (May 2016), pp. 395–465. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2015.11.001.

[47] S. Chu, K. Weitz, A. Cheung, and D. Suciu. “HoTTSQL: Proving Query Rewrites with Univalent SQL Semantics.” In: SIGPLAN Not. 52.6 (June 2017), pp. 510–524. ISSN: 0362-1340. DOI: 10.1145/3140587.3062348.

[48] S. Chu, D. Li, C. Wang, A. Cheung, and D. Suciu. “Demonstration of the Cosette Automated SQL Prover.” In: Proceedings of the 2017 ACM International Conference on Management of Data. SIGMOD ’17. USA: Association for Computing Machinery, 2017, pp. 1591–1594. ISBN: 9781450341974. DOI: 10.1145/3035918.3058728.

[49] S. Chu, D. Li, C. Wang, A. Cheung, D. Suciu, and K. Weitz. Cosette: An Automated SQL Solver. on-line at: http://cosette.cs.washington.edu/, retrieved November 1st, 2020.

[50] S. Chu, D. Li, C. Wang, A. Cheung, D. Suciu, and K. Weitz. GitHub repository: Cosette. on-line at: https://github.com/uwdb/Cosette, retrieved November 1st, 2020.

[51] S. Chu, B. Murphy, J. Roesch, A. Cheung, and D. Suciu. “Axiomatic Foundations and Algorithms for Deciding Semantic Equivalences of SQL Queries.” In: Proc. VLDB Endow. 11.11 (July 2018), pp. 1482–1495. ISSN: 2150-8097. DOI: 10.14778/3236187.3236200.

[52] S. Chu, C. Wang, K. Weitz, and A. Cheung. “Cosette: An Automated Prover for SQL.” In: CIDR. 2017.

[53] E. Clarke, D. Kroening, and F. Lerda. “A Tool for Checking ANSI-C Programs.” In: TACAS. Springer, 2004, pp. 168–176. DOI: 10.1007/978-3-540-24730-2_15.

[54] E. Clarke, D. Kroening, and F. Lerda. “A Tool for Checking ANSI-C Programs.” In: Tools and Algorithms for the Construction and Analysis of Systems. Ed. by K. Jensen and A. Podelski. Springer, 2004, pp. 168–176. ISBN: 978-3-540-24730-2.

[55] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, eds. Handbook of Model Checking. Springer, 2018. ISBN: 978-3-319-10574-1. DOI: 10.1007/978-3-319-10575-8.

[56] S. Cohen. “Equivalence of Queries Combining Set and Bag-set Semantics.” In: PODS. Chicago, IL, USA: ACM, 2006, pp. 70–79. ISBN: 1-59593-318-2. DOI: 10.1145/1142351.1142362.

[57] S. Cohen. “Equivalence of queries that are sensitive to multiplicities.” In: The VLDB Journal 18.3 (June 2009), pp. 765–785. ISSN: 0949-877X. DOI: 10.1007/s00778-008-0122-1.

[58] S. Cohen, W. Nutt, and A. Serebrenik. “Rewriting Aggregate Queries Using Views.” In: PODS. Philadelphia, Pennsylvania, USA: ACM, 1999, pp. 155–166. ISBN: 1-58113-062-7. DOI: 10.1145/303976.303992.

[59] W. W. W. Consortium. XML Path Language (XPath). on-line at: https://www.w3.org/TR/xpath/, retrieved November 1st, 2020.

[60] L. Cordeiro, B. Fischer, and J. Marques-Silva. “SMT-Based Bounded Model Checking for Embedded ANSI-C Software.” In: 2009 IEEE/ACM International Conference on Automated Software Engineering. 2009, pp. 137–148. DOI: 10.1109/ASE.2009.63.

[61] L. Cordeiro, B. Fischer, and J. Marques-Silva. “SMT-Based Bounded Model Checking for Embedded ANSI-C Software.” In: ASE (2009), pp. 137–148. DOI: 10.1109/ASE.2009.63.

[62] L. Cordeiro, D. Kroening, and P. Schrammel. “JBMC: Bounded Model Checking for Java Bytecode.” In: Tools and Algorithms for the Construction and Analysis of Systems. Ed. by D. Beyer, M. Huisman, F. Kordon, and B. Steffen. Springer, 2019, pp. 219–223. ISBN: 978-3-030-17502-3.

[63] C. Coronel, S. Morris, and P. Rob. Database Systems: Design, Implementation, and Management. 9th. USA: Course Technology Press, 2009. ISBN: 0538469684.

[64] P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.” In: POPL. ACM Press, 1977, pp. 238–252. DOI: 10.1145/512950.512973.

[65] F. Darari, W. Nutt, S. Razniewski, and S. Rudolph. “Completeness and soundness guarantees for conjunctive SPARQL queries over RDF data sources with completeness statements1.” In: Semantic Web (Feb. 2019), pp. 1–42. DOI: 10.3233/SW-190344.

[66] C. J. Date. An Introduction to Database Systems (8th edition). Pearson, 2004.

[67] L. De Moura and N. Bjorner. “Z3: An Efficient SMT Solver.” In: TACAS. 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[68] A. Deutsch. Static Verification of Dynamic Properties. White paper, PolySpace Technologies Inc. 2003.

[69] A. Deutsch, L. Popa, and V. Tannen. “Physical Data Independence, Constraints, and Optimization with Universal Plans.” In: Proceedings of the 25th International Conference on Very Large Data Bases. VLDB ’99. Morgan Kaufmann Publishers Inc., 1999, pp. 459–470. ISBN: 1558606157.

[70] G. Dong and J. Su. “Conjunctive Query Containment with Respect to Views and Constraints.” In: Inf. Process. Lett. 57.2 (Jan. 1996), pp. 95–102. ISSN: 0020-0190. DOI: 10.1016/0020-0190(95)00192-1.

[71] F. Donini, M. Lenzerini, and D. Nardi. “AL-log: Integrating Datalog and Description Logics.” In: Journal of Intelligent Information Systems 10 (Dec. 1999). DOI: 10.1023/A:1008687430626.

[72] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. “Reasoning in Description Logics.” In: Center for the Study of Language and Information (June 1999).

[73] G. Douglas and R. Lawrence. “Improving SQL Query Performance on Embedded Devices Using Pre-compilation.” In: SAC. Pisa, Italy: ACM, 2016, pp. 961–964. ISBN: 978-1-4503-3739-7. DOI: 10.1145/2851613.2851657.

[74] M. Duerst and M. Suignard. RFC 3987: Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard). Internet Engineering Task Force, Jan. 2005. URL: http://www.ietf.org/rfc/rfc3987.txt.

[75] N. Ebel. Manually Trigger A GitHub Actions Workflow. on-line at: https://goobar.io/2019/12/07/manually-trigger-a-github-actions-workflow/, retrieved November 1st, 2020.

[76] M. Emmi, R. Majumdar, and K. Sen. “Dynamic Test Input Generation for Database Applications.” In: ISSTA. London, United Kingdom: ACM, 2007, pp. 151–162. ISBN: 978-1-59593-734-6. DOI: 10.1145/1273463.1273484.

[77] D. Felsing, S. Grebing, V. Klebanov, P. Rümmer, and M. Ulbrich. “Automating Regression Verification.” In: Automated Software Engineering. ACM, 2014, pp. 349–360. DOI: 10.1145/2642937.2642987.

[78] M. Fernández, D. Florescu, A. Levy, and D. Suciu. “Verifying Integrity Constraints on Web Sites.” In: vol. 1. Jan. 1999, pp. 614–619.

[79] M. Fisher, J. Ellis, and J. C. Bruce. JDBC API Tutorial and Reference. 3rd ed. Pearson Education, 2003. ISBN: 0321173848.

[80] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata. “Extended Static Checking for Java.” In: SIGPLAN Not. 37.5 (May 2002), pp. 234–245. DOI: 10.1145/543552.512558.

[81] S. Flesca, F. Furfaro, and E. Masciari. “On the Minimization of XPath Queries.” In: J. ACM 55.1 (Feb. 2008). ISSN: 0004-5411. DOI: 10.1145/1326554.1326556.

[82] D. Florescu, A. Levy, and D. Suciu. “Query Containment for Conjunctive Queries with Regular Expressions.” In: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS ’98. Seattle, Washington, USA: ACM, 1998, pp. 139–148. ISBN: 0897919963. DOI: 10.1145/275487.275503.

[83] T. A. S. Foundation. ARQ - A SPARQL Processor for Jena. on-line at: https://jena.apache.org/documentation/query/, retrieved November 1st, 2020.

[84] M. Fowler. Refactoring: improving the design of existing code. Addison-Wesley Professional, 2018.

[85] M. Friedman, A. Levy, and T. Millstein. “Navigational Plans for Data Integration.” In: Proceedings of the 1999 International Conference on Intelligent Information Integration - Volume 23. III’99. Stockholm, Sweden: CEUR-WS.org, 1999, pp. 72–78.

[86] V. Ganesh and D. L. Dill. “A Decision Procedure for Bit-Vectors and Arrays.” In: Computer Aided Verification. Ed. by W. Damm and H. Hermanns. Springer, 2007, pp. 519–531. ISBN: 978-3-540-73368-3.

[87] J. Garbus. SAP ASE 16/Sybase ASE Administration. Sap Press, 2015.

[88] H. Garcia-Molina, J. D. Ullman, and J. Widom. Database systems - the complete book (2. ed.). Pearson Education, 2009, pp. I–XXVI, 1–1203. ISBN: 978-0-13-187325-4.

[89] A. Gaviar. GitHub’s Top 100 Most Valuable Repositories Out of 96 Million. on-line at: https://hackernoon.com/githubs-top-100-most-valuable-repositories-out-of-96-million-bb48caa9eb0b, retrieved November 1st, 2020. 2019.

[90] P. Genevès and N. Layaïda. “A System for the Static Analysis of XPath.” In: ACM Trans. Inf. Syst. 24.4 (Oct. 2006), pp. 475–502. ISSN: 1046-8188. DOI: 10.1145/1185877.1185882.

[91] P. Genevès, N. Layaïda, and A. Schmitt. “Efficient Static Analysis of XML Paths and Types.” In: SIGPLAN Not. 42.6 (June 2007), pp. 342–351. ISSN: 0362-1340. DOI: 10.1145/1273442.1250773.

[92] P. Genevès, N. Layaïda, A. Schmitt, N. Gesbert, and V. Knyttl. The Tree Reasoning Solver. on-line at: http://tyrex.inria.fr/websolver/, retrieved November 1st, 2020.

[93] K. Georgala, M. Spasić, M. Jovanovik, V. Papakonstantinou, C. Stadler, M. Röder, and A.-C. N. Ngomo. “MOCHA2018: The Mighty Storage Challenge at ESWC 2018.” In: Semantic Web Challenges. Ed. by D. Buscaldi, A. Gangemi, and D. Reforgiato Recupero. Springer, 2018, pp. 3–16. ISBN: 978-3-030-00072-1.

[94] M. A. Ghafoor, M. S. Mahmood, and J. H. Siddiqui. “Extending symbolic execution for automated testing of stored procedures.” In: Software Quality Journal (July 2019). ISSN: 1573-1367. DOI: 10.1007/s11219-019-09453-6.

[95] B. Godlin and O. Strichman. “Regression verification: proving the equivalence of similar programs.” In: Softw. Test., Verif. Reliab. 23.3 (2013), pp. 241–258. DOI: 10.1002/stvr.1472.

[96] C. Görg and P. Weiundefinedgerber. “Error Detection by Refactoring Reconstruction.” In: Proceedings of the 2005 International Workshop on Mining Software Repositories. MSR ’05. St. Louis, Missouri: Association for Computing Machinery, 2005, pp. 1–5. ISBN: 1595931236. DOI: 10.1145/1083142.1083148.

[97] J. Groppe and S. Groppe. “A Prototype of a Schema-Based XPath Satisfiability Tester.” In: Database and Expert Systems Applications. Ed. by S. Bressan, J. Küng, and R. Wagner. Springer, 2006, pp. 93–103. ISBN: 978-3-540-37872-3.

[98] S. Grossman, S. Cohen, S. Itzhaky, N. Rinetzky, and M. Sagiv. “Verifying Equivalence of Spark Programs.” In: CAV. LNCS. Springer, 2017, pp. 282–300. DOI: 10.1007/978-3-319-63390-9_15.

[99] P. Guagliardo and L. Libkin. “A Formal Semantics of SQL Queries, Its Validation, and Applications.” In: Proc. VLDB Endow. 11.1 (Sept. 2017), pp. 27–39. ISSN: 2150-8097. DOI: 10.14778/3151113.3151116.

[100] A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. “Constraint Checking with Partial Information.” In: Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS ’94. Minneapolis, Minnesota, USA: ACM, 1994, pp. 45–55. ISBN: 0897916425. DOI: 10.1145/182591.182597.

[101] A. Gupta and J. D. Ullman. “Generalizing Conjunctive Query Containment for View Maintenance and Integrity Constraint Verification (Abstract).” In: Proceedings of the Workshop on Deductive Databases held in conjunction with the Joint International Conference and Symposium on Logic Programming. Ed. by K. Ramamohanarao, J. Harland, and G. Dong. Vol. CITRI/TR-92-65. Technical Report. Department of Computer Science, University of Melbourne, 1992, p. 195.

[102] P. Hanson and M. Mani. “Semantic Optimization of XQuery by Rewriting.” In: Proceedings of Advances in Databases and Information Systems. Ed. by J. Grundspenkis, M. Kirikova, Y. Manolopoulos, and L. Novickis. Springer, 2010, pp. 87–95. ISBN: 978-3-642-12082-4.

[103] R. Hayek and O. Shmueli. “Improved Cardinality Estimation by Learning Queries Containment Rates.” In: arXiv preprint arXiv:1908.07723 (2019).

[104] P. Hitzler, M. Krotzsch, and S. Rudolph. Foundations of Semantic Web Technologies. Chapman and Hall/CRC, Aug. 2009. DOI: 10.1201/9781420090512.

[105] C. A. R. Hoare. “An Axiomatic Basis for Computer Programming.” In: Commun. ACM 12.10 (Oct. 1969), pp. 576–580. DOI: 10.1145/363235.363259.

[106] A. Hogan. “The Semantic Web: Two decades on.” In: Semantic Web 11 (Jan. 2020), pp. 169–185. DOI: 10.3233/SW-190387.

[107] S. Huang and K. Cheng. Formal equivalence checking and design debugging. Kluwer Academic Publishers, 1998. DOI: 10.1007/978-1-4615-5693-0.

[108] IBM. Developing Embedded SQL Applications. on-line at: ftp://ftp.software.ibm.com/ps/products/db2/info/vr101/pdf/en_US/DB2DevEmbeddedSQL-db2a1e1010.pdf, retrieved November 1st, 2020.

[109] Y. Ioannidis and R. Ramakrishnan. “Containment of Conjunctive Queries: Beyond Relations as Sets.” In: ACM Trans. Database Syst. 20.3 (1995), pp. 288–324. DOI: 10.1145/211414.211419.

[110] ISO. ISO/IEC 9075-15:2019. on-line at: https://www.iso.org/standard/67382.html, retrieved November 1st, 2020.

[111] ITTIA. Benefits of Database for Embedded System and IoT Device Manufacturers. online at: http://www.ittia.com/files/Benefits_of_Database_for_Embedded_System_and_IoT_Device_Manufacturers.pdf, retrieved November 1st, 2020.

[112] T. S. Jayram, P. G. Kolaitis, and E. Vee. “The Containment Problem for Real Conjunctive Queries with Inequalities.” In: PODS. ACM, 2006, pp. 80–89. ISBN: 1-59593-318-2. DOI: 10.1145/1142351.1142363.

[113] D. Johnson and A. Klug. “Testing containment of conjunctive queries under functional and inclusion dependencies.” In: Journal of Computer and System Sciences 28.1 (1984), pp. 167–189. ISSN: 0022-0000. DOI: 10.1016/0022-0000(84)90081-3.

[114] M. Jovanovik and M. Spasić. "Benchmarking Virtuoso 8 at the Mighty Storage Challenge 2018: Challenge Results." In: Semantic Web Challenges - 5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, June 3-7, 2018, Revised Selected Papers. Ed. by D. Buscaldi, A. Gangemi, and D. R. Recupero. Vol. 927. Communications in Computer and Information Science. Springer, 2018, pp. 24–35. DOI: 10.1007/978-3-030-00072-1_3.

[115] V. Kashyap, C. Bussler, and M. Moran. The Semantic Web. Springer, Aug. 15, 2008. 432 pp. ISBN: 3540764518.

[116] M. Kim, T. Zimmermann, and N. Nagappan. "An Empirical Study of Refactoring Challenges and Benefits at Microsoft." In: IEEE Trans. Softw. Eng. 40.7 (July 2014), pp. 633–649. ISSN: 0098-5589. DOI: 10.1109/TSE.2014.2318734. URL: https://doi.org/10.1109/TSE.2014.2318734.

[117] J. C. King. "Symbolic Execution and Program Testing." In: Communications of the ACM 19.7 (1976), pp. 385–394. DOI: 10.1145/360248.360252.

[118] A. Klug. "On Conjunctive Queries Containing Inequalities." In: J. ACM 35.1 (Jan. 1988), pp. 146–160. ISSN: 0004-5411. DOI: 10.1145/42267.42273.

[119] G. Klyne, J. Carroll, and B. McBride. Resource Description Framework: Concepts and Abstract Syntax. Jan. 2004. URL: https://www.w3.org/TR/rdf-concepts/.

[120] E. Kostylev, J. Reutter, and D. Vrgoč. "Containment of queries for graphs with data." In: Journal of Computer and System Sciences 92 (Sept. 2017). DOI: 10.1016/j.jcss.2017.09.005.

[121] E. V. Kostylev, J. L. Reutter, and D. Vrgoč. "Static analysis of navigational XPath over graph databases." In: Information Processing Letters 116.7 (2016), pp. 467–474. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2016.03.006.

[122] D. Kozen. "Results on the Propositional μ-Calculus." In: Proceedings of the 9th Colloquium on Automata, Languages and Programming. Springer, 1982, pp. 348–359. ISBN: 3-540-11576-5. URL: http://dl.acm.org/citation.cfm?id=646236.682866.

[123] O. Lassila and R. R. Swick. Resource Description Framework (RDF): Model and Syntax Specification. 1999. URL: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[124] C. Lattner and V. Adve. "LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation." In: CGO. IEEE Computer Society, 2004, pp. 75–86. ISBN: 0-7695-2102-9. DOI: 10.1109/CGO.2004.1281665.

[125] A. Letelier, J. Pérez, R. Pichler, and S. Skritek. "Static Analysis and Optimization of Semantic Web Queries." In: ACM Trans. Database Syst. 38.4 (Dec. 2013), 25:1–25:45. ISSN: 0362-5915. DOI: 10.1145/2500130.

[126] A. Letelier, J. Pérez, R. Pichler, and S. Skritek. "Static analysis and optimization of semantic web queries." In: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012. Ed. by M. Benedikt, M. Krötzsch, and M. Lenzerini. ACM, 2012, pp. 89–100. DOI: 10.1145/2213556.2213572.

[127] A. Y. Levy, A. Rajaraman, and J. J. Ordille. "Query-Answering Algorithms for Information Agents." In: Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1. AAAI'96. Portland, Oregon: AAAI Press, 1996, pp. 40–47. ISBN: 026251091X.

[128] A. Y. Levy and M.-C. Rousset. “Combining Horn rules and description logics in CARIN.” In: Artificial Intelligence 104.1 (1998), pp. 165–209. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00048-4.

[129] A. Y. Levy and D. Suciu. “Deciding Containment for Queries with Complex Objects and Aggregations.” In: PODS 1997. 1997.

[130] M.Marcozzi, W.Vanhoof, and J.L.Hainaut. “Relational symbolic exec. of SQL code for unit testing of database programs.” In: Sci. of Comp. Program. 105 (2015), pp. 44–72. ISSN: 0167-6423. DOI: 10.1016/j.scico.2015.03.005.

[131] M. S. Mahmood, M. Ghafoor, and J. H. Siddiqui. “Symbolic Execution of Stored Procedures in Database Management Systems.” In: ASE. ACM, 2016, pp. 519–530. ISBN: 978-1-4503-3845-5. DOI: 10.1145/2970276.2970318.

[132] G. Malecha, G. Morrisett, A. Shinnar, and R. Wisnesky. “Toward a Verified Relational DB Management System.” In: POPL. ACM, 2010, pp. 237–248. ISBN: 978-1-60558-479-9. DOI: 10.1145/1706299.1706329.

[133] D. Marker. Model Theory : An Introduction. Graduate Texts in Mathematics. Springer, 2002. ISBN: 9780387987606. URL: https://books.google.rs/books?id=QieAHk--GCcC.

[134] J. P. McCrae. The Linked Open Data Cloud. Insight Centre for Data Analytics. URL: https://lod-cloud.net/.

[135] T. Mens and T. Tourwé. “A Survey of Software Refactoring.” In: IEEE Transactions on Software Engineering 30.2 (Feb. 2004), pp. 126–139. ISSN: 0098-5589. DOI: 10.1109/TSE.2004.1265817.

[136] F. Merz, S. Falke, and C. Sinz. “LLBMC: Bounded Model Checking of C and C++ Programs Using a Compiler IR.” In: VSTTE. LNCS. Springer, 2012, pp. 146–161. DOI: 10.1007/978-3-642-27705-4_12.

[137] F. Merz, S. Falke, and C. Sinz. “LLBMC: Bounded Model Checking of C and C++ Programs Using a Compiler IR.” In: Verified Software: Theories, Tools, Experiments. Ed. by R. Joshi, P. Müller, and A. Podelski. Springer, 2012, pp. 146–161. ISBN: 978-3-642-27705-4.

[138] R. van der Meyden. “The Complexity of Querying Indefinite Information: Defined Relations, Recursion and Linear Order.” PhD thesis. USA: Rutgers University, 1992.

[139] Microsoft. Microsoft SQL Server. on-line at: https://www.microsoft.com/en-us/sql-server/, retrieved November 1st, 2020. 2020.

[140] G. Miklau and D. Suciu. “Containment and Equivalence for a Fragment of XPath.” In: J. ACM 51.1 (Jan. 2004), pp. 2–45. ISSN: 0004-5411. DOI: 10.1145/962446.962448.

[141] G. Miklau and D. Suciu. “Containment and Equivalence for an XPath Fragment.” In: June 2002, pp. 65–76. DOI: 10.1145/543613.543623.

[142] L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. “The Lean Theorem Prover (System Description).” In: Automated Deduction - CADE-25. Ed. by A. P. Felty and A. Middeldorp. Springer, 2015, pp. 378–388. ISBN: 978-3-319-21401-6.

[143] E. Murphy-Hill, C. Parnin, and A. P. Black. “How We Refactor, and How We Know It.” In: IEEE Transactions on Software Engineering 38.1 (Jan. 2012), pp. 5–18. ISSN: 2326-3881. DOI: 10.1109/TSE.2011.41.

[144] M. Negri, G. Pelagatti, and L. Sbattella. “Formal Semantics of SQL Queries.” In: ACM Trans. Database Syst. 16.3 (Sept. 1991), pp. 513–534. ISSN: 0362-5915. DOI: 10.1145/111197.111212.

[145] R. O. Obe and L. S. Hsu. PostgreSQL: Up and Running: a Practical Guide to the Advanced Open Source Database. “ O’Reilly Media, Inc.”, 2017.

[146] Oracle. Oracle Database. on-line at: https://www.oracle.com/database/, retrieved November 1st, 2020. 2020.

[147] K. Pan, X. Wu, and T. Xie. “Guided Test Generation for Database Applications via Synthesized Database Interactions.” In: ACM Trans. Softw. Eng. Methodol. 23.2 (Apr. 2014), 12:1–12:27. ISSN: 1049-331X. DOI: 10.1145/2491529.

[148] J. Pérez, M. Arenas, and C. Gutierrez. “Semantics and Complexity of SPARQL.” In: The Semantic Web - ISWC 2006. Ed. by I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. M. Aroyo. Springer, 2006, pp. 30–43. ISBN: 978-3-540-49055-5.

[149] J. Pérez, M. Arenas, and C. Gutierrez. “Semantics and Complexity of SPARQL.” In: ACM Trans. Database Syst. 34.3 (Sept. 2009), 16:1–16:45. ISSN: 0362-5915. DOI: 10.1145/1567274.1567278.

[150] F. Picalausa and S. Vansummeren. “What Are Real SPARQL Queries Like?” In: Proceedings of the International Workshop on Semantic Web Information Management. SWIM ’11. Athens, Greece: ACM, 2011. ISBN: 9781450306515. DOI: 10.1145/1999299.1999306.

[151] R. Pichler and S. Skritek. “Containment and Equivalence of Well-Designed SPARQL.” In: Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. PODS ’14. Snowbird, Utah, USA: ACM, 2014, pp. 39–50. ISBN: 9781450323758. DOI: 10.1145/2594538.2594542.

[152] A. Polleres, J. Reutter, and E. Kostylev. “Nested constructs vs. sub-selects in SPARQL.” In: vol. 1644. CEUR Workshop Proceedings, 2016.

[153] H. Post and C. Sinz. “Proving Functional Equivalence of Two AES Implementations Using Bounded Model Checking.” In: ICST 2009. 2009, pp. 31–40. DOI: 10.1109/ICST.2009.39.

[154] PostgreSQL. PostgreSQL: The World’s Most Advanced Open Source Relational Database. on-line at: https://www.postgresql.org/, retrieved November 1st, 2020. 2020.

[155] E. Prud’hommeaux and A. Seaborne. SPARQL 1.1 Query Language. on-line at: https://www.w3.org/TR/sparql11-query/, retrieved November 1st, 2020. 2013.

[156] E. Prud’hommeaux and A. Seaborne. SPARQL Query Language for RDF, W3C Recommendation. on-line at: http://www.w3.org/TR/rdf-sparql-query/, retrieved November 1st, 2020. 2008.

[157] D. A. Ramos and D. R. Engler. “Practical, Low-Effort Equivalence Verification of Real Code.” In: CAV. 2011.

[158] A. Riazanov and A. Voronkov. “The design and implementation of VAMPIRE.” In: AI Commun. 15.2-3 (2002), pp. 91–110.

[159] G. Rull, P. A. Bernstein, I. G. dos Santos, Y. Katsis, S. Melnik, and E. Teniente. "Query Containment in Entity SQL." In: International Conference on Management of Data. SIGMOD '13. New York, New York, USA: ACM, 2013, pp. 1169–1172. ISBN: 9781450320375. DOI: 10.1145/2463676.2463711.

[160] Y. Sagiv and M. Yannakakis. "Equivalences Among Relational Exp. with Union and Difference Operators." In: J. ACM 27.4 (Oct. 1980), pp. 633–655. ISSN: 0004-5411. DOI: 10.1145/322217.322221.

[161] M. Saleem, C. Stadler, Q. Mehmood, J. Lehmann, and A.-C. N. Ngomo. "SQCFramework: Generating SPARQL Query Containment Benchmarks using the SQCFramework." In: Proceedings of the 17th International Semantic Web Conference (ISWC), 2018 - Posters & Demos. 2018.

[162] M. Saleem, C. Stadler, Q. Mehmood, J. Lehmann, and A.-C. N. Ngomo. "SQCFramework: SPARQL Query Containment Benchmark Generation Framework." In: Proceedings of the Knowledge Capture Conference. K-CAP 2017. Austin, TX, USA: ACM, 2017, 28:1–28:8. ISBN: 978-1-4503-5553-7. DOI: 10.1145/3148011.3148017.

[163] ScaleGrid. Database Trends - SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use. on-line at: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/, retrieved November 1st, 2020.

[164] C. Scheben and P. H. Schmitt. "Efficient Self-composition for Weakest Precondition Calculi." English. In: FM. Vol. 8442. LNCS. Springer, 2014, pp. 579–594. DOI: 10.1007/978-3-319-06410-9_39.

[165] M. Schlaipfer, K. Rajan, A. Lal, and M. Samak. "Optimizing Big-Data Queries Using Program Synthesis." In: SOSP. Shanghai, China: ACM, 2017, pp. 631–646. ISBN: 978-1-4503-5085-3. DOI: 10.1145/3132747.3132773.

[166] M. Schmidt, M. Meier, and G. Lausen. "Foundations of SPARQL Query Optimization." In: Proceedings of the 13th International Conference on Database Theory. ICDT '10. Lausanne, Switzerland: ACM, 2010, pp. 4–33. ISBN: 9781605589473. DOI: 10.1145/1804669.1804675.

[167] D. Schmitt. Bug #5673: Optimizer creates strange execution plan leading to wrong results. on-line at: https://www.postgresql.org/message-id/201009231503.o8NF3Blt059661@wwwmaster.postgresql.org, retrieved November 1st, 2020.

[168] T. Schwentick. "XPath Query Containment." In: SIGMOD Rec. 33.1 (Mar. 2004), pp. 101–109. ISSN: 0163-5808. DOI: 10.1145/974121.974140.

[169] G. Serfiotis, I. Koffina, V. Christophides, and V. Tannen. "Containment and Minimization of RDF/S Query Patterns 1." In: Nov. 2005, pp. 607–623. DOI: 10.1007/11574620_44.

[170] M. Smid. "OnGIS: Geospatial Data Integration Using Semantic Technologies and Query Containment." In: Int. J. Semant. Web Inf. Syst. 15.1 (Jan. 2019), pp. 1–20. ISSN: 1552-6283. DOI: 10.4018/IJSWIS.2019010101.

[171] M. Spasić and M. Vujošević Janičić. "First steps towards proving functional equivalence of embedded SQL." In: TYPES. Univ. Minho, 2018, pp. 78–79.

[172] M. Spasić and M. Vujošević Janičić. "SpeCS — SPARQL Query Containment Solver." In: 2020 Zooming Innovation in Consumer Technologies Conference (ZINC). IEEE, 2020, pp. 31–35. DOI: 10.1109/ZINC50678.2020.9161435.

[173] M. Spasić and M. Vujošević Janičić. "Verification supported refactoring of embedded SQL." In: Software Quality Journal (June 2020), pp. 1–37. DOI: 10.1007/s11219-020-09517-y.

[174] M. Spasić. "Linked Data Lifecycle of Floating Car Data." In: InfoM - Journal of Information Technology and Multimedia Syst. 71 (2020), pp. 44–50. ISSN: 1451-4397.

[175] M. Spasić and M. Jovanovik. "MOCHA 2017 as a Challenge for Virtuoso." In: Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017, Revised Selected Papers. Ed. by M. Dragoni, M. Solanki, and E. Blomqvist. Vol. 769. Communications in Computer and Information Science. Springer, 2017, pp. 21–32. DOI: 10.1007/978-3-319-69146-6_3.

[176] M. Spasić, M. Jovanovik, and A. Prat-Pérez. "An RDF Dataset Generator for the Social Network Benchmark with Real-World Coherence." In: Proceedings of the Workshop on Benchmarking Linked Data (BLINK 2016) co-located with the 15th International Semantic Web Conference (ISWC), Kobe, Japan, October 18, 2016. Ed. by I. Fundulaki, A. Krithara, A.-C. N. Ngomo, and V. Rentoumi. Vol. 1700. CEUR Workshop Proceedings. Oct. 2016. DOI: 10.13140/RG.2.2.30490.64965/1.

[177] M. Spasić and M. Vujošević Janičić. GitHub repository: SQLC. on-line at: https://github.com/mirkospasic/sqlc, retrieved November 1st, 2020.

[178] M. Spasić and M. Vujošević Janičić. SpeCS Web page. on-line at: http://argo.matf.bg.ac.rs/?content=specs, retrieved November 1st, 2020. 2020.

[179] C. Stadler. Query Containment Module. URL: https://github.com/AKSW/jena-sparql-api/tree/develop/jena-sparql-api-query-containment.

[180] C. Stadler, M. Saleem, A.-C. N. Ngomo, and J. Lehmann. "Efficiently Pinpointing SPARQL Query Containments." In: Web Engineering. Ed. by T. Mikkonen, R. Klamma, and J. Hernández. Springer, 2018, pp. 210–224. ISBN: 978-3-319-91662-0.

[181] O. Strichman and B. Godlin. "Regression Verification - A Practical Way to Verify Programs." In: Verified Software: Theories, Tools, Experiments: First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10-13, 2005, Revised Selected Papers and Discussions. Ed. by B. Meyer and J. Woodcock. Springer, 2008, pp. 496–501. ISBN: 978-3-540-69149-5. DOI: 10.1007/978-3-540-69149-5_54.

[182] M. Sulik. Bug #70038: Wrong select count distinct with a field included in two-column unique key. on-line at: https://bugs.mysql.com/bug.php?id=70038, retrieved November 1st, 2020.

[183] S. Sybase. Relational Database Server. on-line at: https://www.sap.com/products/sybase-ase.html, retrieved November 1st, 2020. 2020.

[184] R. Taelman, P. Colpaert, E. Mannens, and R. Verborgh. "Generating public transport data based on population distributions for RDF benchmarking." In: Semantic Web 10 (Jan. 2019), pp. 305–328. DOI: 10.3233/SW-180319.

[185] Y. Tanabe, K. Takahashi, M. Yamamoto, A. Tozawa, and M. Hagiya. "A Decision Procedure for the Alternation-Free Two-Way Modal μ-Calculus." In: Automated Reasoning with Analytic Tableaux and Related Methods. Ed. by B. Beckert. Springer, 2005, pp. 277–291. ISBN: 978-3-540-31822-4.

[186] N. Tillmann and J. de Halleux. “Pex–White Box Test Generation for .NET.” In: Tests and Proofs. Ed. by B. Beckert and R. Hähnle. Vol. 4966. LNCS. Springer, 2008, pp. 134–153. ISBN: 978-3-540-79124-9. DOI: 10.1007/978-3-540-79124-9_10.

[187] B. A. Trakhtenbrot. “Impossibility of an algorithm for the decision problem in finite classes.” In: Doklady Akademii Nauk SSSR 70 (1950), pp. 569–572.

[188] J. D. Ullman. “Information integration using logical views.” In: Theoretical Computer Science 239.2 (2000), pp. 189–210. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(99)00219-4.

[189] M. Veanes, N. Tillmann, and J. de Halleux. “Qex: Symbolic SQL Query Explorer.” In: LPAR. Springer, 2010, pp. 425–446. ISBN: 978-3-642-17511-4. DOI: 10.1007/978-3-642-17511-4_24.

[190] S. Verdoolaege, M. Palkovic, M. Bruynooghe, G. Janssens, and F. Catthoor. “Experience with Widening Based Equivalence Checking in Realistic Multimedia Systems.” In: J. Electronic Testing 26.2 (2010), pp. 279–292. DOI: 10.1007/s10836-009-5140-4.

[191] M. Vujošević Janičić. “Concurrent Bug Finding Based on Bounded Model Checking.” In: International Journal of Software Engineering and Knowledge Engineering 30.5 (2020), pp. 669–694. DOI: 10.1142/S0218194020500242.

[192] M. Vujošević Janičić. “Regression verification by system LAV.” In: InfoM - Journal of Information Technology and Multimedia Syst. 49 (2014), pp. 14–20.

[193] M. Vujošević Janičić and V. Kuncak. “Development and Evaluation of LAV: An SMT-Based Error Finding Platform.” In: VSTTE. LNCS. Springer, 2012, pp. 98–113. DOI: 10.1007/978-3-642-27705-4_9.

[194] M. Vujošević Janičić and F. Marić. “Regression verification for automated evaluation of students programs.” In: Computer Science and Information Systems 17.1 (2020), pp. 205–228. DOI: https://doi.org/10.2298/CSIS181220019V.

[195] M. Vujošević Janičić, M. Nikolić, D. Tošić, and V. Kuncak. “Software verification and graph similarity for automated evaluation of students’ assignments.” In: Information and Softw. Technology 55.6 (2013). DOI: 10.1016/j.infsof.2012.12.005.

[196] M. Vujošević Janičić and M. Spasić. Tools LAV and SQLAV. on-line at: http://argo.matf.bg.ac.rs/?content=lav, retrieved November 1st, 2020.

[197] Y. Wang, I. Dillig, S. Lahiri, and W. Cook. “Verifying Equivalence of Database-Driven Apps.” In: Proc. ACM Program. Lang. (Jan. 2017), 56:1–56:29. ISSN: 2475-1421. DOI: 10.1145/3158144.

[198] L. Web. Ten Ways Databases Run Your Life. on-line at: https://www.liquidweb.com/blog/ten-ways-databases-run-your-life/, retrieved November 1st, 2020. 2020.

[199] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. “SPASS Version 3.5.” In: CADE. 2009, pp. 140–145. DOI: 10.1007/978-3-642-02959-2_10.

[200] P. Weißgerber and S. Diehl. “Are refactorings less error-prone than other changes?” In: Proceedings of the 2006 international workshop on Mining software repositories. Jan. 2006, pp. 112–118. DOI: 10.1145/1137983.1138011.

[201] P. Weißgerber and S. Diehl. “Identifying Refactorings from Source-Code Changes.” In: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering. ASE ’06. USA: IEEE Computer Society, 2006, pp. 231–240. ISBN: 0769525792. DOI: 10.1109/ASE.2006.41.

[202] Y. Welsch and A. Poetzsch-Heffter. “A fully abstract trace-based semantics for reasoning about backward compatibility of class libraries.” In: Sci. Comput. Program. 92 (2014), pp. 129–161. DOI: 10.1016/j.scico.2013.10.002.

[203] M. Wudage Chekol, J. Euzenat, P. Genevès, and N. Layaïda. “Evaluating and Benchmarking SPARQL Query Containment Solvers.” In: Proceedings of the Semantic Web – ISWC 2013. Ed. by H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz. Springer, 2013, pp. 408–423. ISBN: 978-3-642-41338-4.

[204] P. Yang, P. Jin, and L. Yue. “Exploiting the Performance-Energy Tradeoffs for Mobile Database Apps.” In: Journal of Universal Comp. Science 20.10 (2014), pp. 1488–1498.

[205] Q. Zhou, J. Arulraj, S. Navathe, W. Harris, and J. Wu. SPES: A Two-Stage Query Equivalence Verifier. 2020. arXiv: 2004.00481 [cs.DB].

[206] Q. Zhou, J. Arulraj, S. Navathe, W. Harris, and D. Xu. “Automated Verification of Query Equivalence Using Satisfiability Modulo Theories.” In: Proceedings of the VLDB Endowment 12.11 (2019), pp. 1276–1288. DOI: 10.14778/3342263.3342267.

Биографија аутора

Мирко Д. Спасић је рођен у Брусу, 01.03.1985. године, где је завршио основну школу „Јован Јовановић Змај" и „Гимназију Брус" као носилац Вукове дипломе и Ђак генерације. Током свог основног и средњег образовања активно учествује на такмичењима из математике на којима осваја бројне награде. Математички факултет у Београду, смер Рачунарство и информатика, уписао је 2004. године и завршио 2009. године први у генерацији, са просечном оценом 10,00 и похвалама Факултета за постигнуте успехе. У току студија био је стипендиста Републичке фондације за развој научног и уметничког подмлатка, као и компаније „Дунав осигурање". На Математичком факултету уписује и докторске студије, у оквиру којих је положио све испите предвиђене планом студија са максималним оценама. За то време, добио је главне награде за најефикаснији систем на такмичењима *MOCHA2017* и *MOCHA2018* у оквиру конференција *ESWC2017* (Хераклион, Грчка) и *ESWC2018* (Порторож, Словенија), и похађао је Летњу школу повезаних отворених података *ISSLOD2011* у Лајпцигу, Немачка.

Запослен је на Катедри за рачунарство и информатику Математичног факултета Универзитета у Београду у звању сарадника у настави од 2009. године, а у звању асистента од 2011. године. Држао је вежбе из следећих предмета: Програмирање 1 и 2, Лексичка анализа са применама, Компилација програмских језика, Превођење програмских језика, Конструкција компилатора, Анализа и дизајн алгоритама, Конструкција и анализа алгоритама 1 и 2. У периоду од 2011. до 2013. године, ангажован је као спољни сарадник у Институту Михајло Пупин, а после тога у компанији *Openlink Software* (Велика Британија), где је радио на многобројним истраживачким FP7 и H2020 пројектима (*LOD2*, *Cesar*, *GeoKnow*, *LDBC*, *Hobbit* и *Sage*). Учесник је пројекта Министарства просвете, науке и технолошког развоја Републике Србије број 178006 под називом „Српски језик и његови ресурси".

Образац 5.

Изјава о ауторству

Име и презиме аутора Мирко Спасић

Број индекса 2031/2016

Изјављујем

да је докторска дисертација под насловом

Моделовање упитних језика са применама у рефакторисању и оптимизацији кода

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, ____.___.2021.

Образац 6.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Мирко Спасић

Број индекса 2031/2016

Студијски програм Информатика

Наслов рада Моделовање упитних језика са применама у рефакторисању и оптимизацији кода

Ментор др. Милена Вујошевић Јаничић

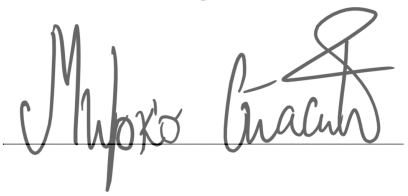
Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, _____.___.2021.



Образац 7.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Моделовање упитних језика са применама у рефакторисању и оптимизацији кода

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, _____.___.2021._____

1. **Ауторство**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.