УДК 004.94

О. Захарова

ВИЗНАЧЕННЯ ТА ВИРІШЕННЯ ЗАДАЧІ ВИЯВЛЕННЯ ВЕБ-СЕРВІСІВ ЗА ДОПОМОГОЮ АПАРАТУ ДЕСКРИПТИВНИХ ЛОГІК

На сьогодні Веб-сервіси дозволяють вирішувати конкретні бізнес-задачі, що реалізують бізнес процеси у різних галузях життєдіяльності людини. Але для того, щоб отримати виконуваний Веб-сервіс, треба вміти ефективно вирішувати цілу низьку задач самих Веб-сервісів на всіх етапах їх життєвого циклу. Апарат дескриптивних логік, завдяки своїм механізмам міркувань та можливостям логічного виводу та надання описам семантичного змісту, є ефективним та потужним інструментом для вирішення задач Веб-сервісів. Мета данної роботи полягає у визначенні ланцюжка задач Веб-сервісів на функціональному рівні та підходів до вирішення цих задач із застосуванням апарата дескриптивних логік.

Ключові слова: семантичний Веб-сервіс, дескриптивна логіка, задача виявлення, пошук веб-сервісу, композиція веб-сервісів, семантичний опис, онтологія домену, онтологія сервісу.

Вступ

Згідно визначення W3C [1–5], під *сервісом* розуміють таку програмну систему, що ідентифікується URI, публічні інтерфейси, прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою Інтернет-Протоколів. Веб-сервіси базуються на чотирьох ключових технологіях [5–7]: eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) та Universal Description, Discovery and Integration (UDDI). Ці технології використовуються для забезпечення функціонування Веб-сервісів.

Хоча Веб-сервіси базуються на широко прийнятих стандартах, відсутність формального опису значень їх функціональності та обміну даних є значною перешкодою у реалізації інтеграційних перспектив. Так як кількість Веб-сервісів зростає, важливо мати автоматизовані засоби для виявлення та композиції Веб-сервісів. Ступінь опису, що є доступною в існуючому стандарті WSDL, залишає місце для неоднозначних інтерпретацій функціональності та даних Веб-сервісів. Неоднозначність інтерпретації ускладнює автоматизацію таких задач, як виявлення, композиція, виклик сервісу тощо. Семантичні описи таких Веб-сервісів відкривають шлях до автоматизації їх композиції.

На практиці існує два рівні представлення сервісів, а саме: функціональна та процесна модель сервісів [8]. На функціональному рівні сервіси розглядаються як окремі існуючі в мережі прикладні компоненти, які можуть бути викликані шляхом відправлення повідомлення. При цьому сервіс виконує свою задачу та (в деяких випадках) виробляє відповідь тому, хто його викликав. Таким чином, відсутня безперервна взаємодія між запитувачем сервісу та самим сервісом. Опис таких Веб-сервісів фокусується на їх функціональності в термінах імені сервіса, імен операцій, імен повідомлень (що відомі також як вхідні та вихідні повідомлення/параметри), імені інтерфейсу.

Сервіси процесного рівня містять декілька операцій, які слідують загальній поведінці сервіса. Такі сервіси потребують розширеної взаємодії між запитувачем сервісу та множиною операцій, забезпечуючи конкретну функціональність. Таким чином, це, як правило, композитні сервіси, тобто взаємодія з ними складається не лише з окремого крока запит-відповідь, для досягнення потрібного результату вони повинні слідувати складному протоколу. Ці кроки можуть складатися з довільних (умовних та ітеративних) комбінацій з

умовними виходами. На цьому рівні необхідний деталізований опис внутрішньої поведінки сервісів, наприклад, за допомогою (STS). Очевидно, що на процесному рівні Веб-сервіси також мають функціональний опис своїх операцій та сервісів. Але сервіси на функціональному рівні не мають поведінкових взаємодій. Це є головною відмінністю цих двох моделей.

Предметом розгляду данної статті є вирішення задачі виявлення для онтологічно анотованих Веб-сервісів, що представлені функціональною моделлю, із застосуванням апарату дескриптивної логіки (ДЛ).

Визначення семантичного Веб-сервісу

ДЛ – формальна мова, яка підтримує концепцію відкритого світу та власні механізми міркувань. Все це робить її бажаною та ефективною як для представлення функціональної частини опису Веб-сервісу, так і для представлення семантичних елементів (анотацій) у процедурному описі Веб-сервісу. А така властивість ДЛ, як підтримка можливостей логічного виведення, забезпечує ефективність вирішення багатьох задач Веб-сервісів, як елементів загальної складної задачі композиції. Так, наприклад, при анотуванні сервісу може виникнути потреба у розширенні загальної онтології домена чи онтології домена конкретного сервісу. Резонери ДЛ дозволять перевірити коректність доданих аксіом. Це зводиться до стандартної задачі виконуваності, що вирішується для цих аксіом.

Онтологія домена

Відповідно до понять ДЛ [5], в межах загальноприйнятої онтології існує термінологічний компонент (T-BOX) [6] і стверджувальний (assertional) компонент (A-BOX). Вважаємо, що T-BOX є єдиним для всіх сервісів, які необхідно анотувати в домені. Він містить всі концепти, які необхідно представляти в домені застосування. Якщо анотації винести до окремого файлу (із посиланнями на файл процесу), то це дозволить глобальні твердження, які дійсні для всіх станів процесу, а саме твердження T-BOX, не повторювати в описі кожного стану процесу, а визначити однократно у відповідному розділі файлу анотацій. A-BOX містить визначення твердження двох різних типів: твердження концептів та твердження ролі. З кожним станом кожного сервісу пов'язується свій A-BOX. Він описує наслідки даної дії в термінах тверджень концепту і ролі. Тим не менш, є деякі твердження, які не залежать від будь-яких дій, але виконуються скрізь. Такі твердження завжди є правильними.

Розглянемо як приклад задачу бронювання квітків на літак. Призначення сервісу – визначити прийнятний до запиту користувача рейс. Входами сервісу є ім'я клієнта, дата відправлення, пункт відправлення та призначення. Виходом є квіток на літак. Онтологія домена буде описана засобами ДЛ. Анотації входів, виходів, передумов та ефектів сервісу є аксіомами визначення або аксіомами включення ДЛ. У WSDL-описі сервісу вони будуть представлені посиланнями на концепти онтології домену *POOntology*.

Спочатку продемонструємо використання апарату ДЛ для визначення онтології домену *POOntology*. Для цього необхідно визначити TBox та ABox ПО.

Відповідний узагальнений TBox може містити такі твердження: *Year, Month, Day, Date, Client, Status, Location, Country, FlightID*

$Date \sqsubseteq has.Year;$

$Date \sqsubseteq has.Month;$

$Date \sqsubseteq has.Day;$

$Location \sqsubseteq isLocatedIn.Country;\ Trip \sqsubseteq hasStartPoint.Location;$

$Trip \sqsubseteq hasEndPoint.Location;\ Trip \sqsubseteq hasDeparture.Date$

$Trip \sqsubseteq hasArrival.Date;\ Flight \sqsubseteq belonguesTo.AirLineRoute$

$Flight \sqsubseteq hasDepartur Date.Date;\ Flight \sqsubseteq hasDepartur Time.Time$

$Flight \sqsubseteq from.Location;\ Flight \sqsubseteq to.Location$

$Flight \sqsubseteq hasSeats.NUMBER;\ Flight \sqsubseteq hasStatus.Status;$

Status = {Available, NotAvailable, booked}

Роль *hasStatus*, що зв'язує два концепти *Flight* та *Status*, фіксує поточний стан запиту клієнта і може приймати наступні значення: *Available* – якщо поїздка доступна; *NotAvailable* – коли поїздка не доступна, *booked* – квиток заброньований.

Додамо передумову до нашого сервісу. Припустимо, що сервіс виконується лише при бронювання квитків до країн Європи. Ефектом є бронювання квитка на літак. Визначимо відповідні умови у TBox, щоб уможливити їх використання. Для цього додамо до TBox концепт *Europe*, а до ABox множину індивидів, які будуть визначати цей клас.

Europe

Ukraine:Europe, France:Europe,

Italy:Europe, Lituva:Europe,

Hungary:Europe, etc.

EuropeanTrip⊑ hasStartPoint Europe ⊓

hasEndPoint Europe

FlightBooked ≡ hasStatus.{booked}

EuropeanTrip ⊑Trip

Семантичне анотування сервісу на функціональному рівні

Для вирішення задач Веб-сервісів їх опис має спиратися на існуючі стандарти. Функціональна модель сервісу (його профілі), що зберігаються в UDDI, визначаються у WSDL, а семантизованих Веб-сервісів, опис яких розширений анотаціями, у розширені WSDL, як, наприклад WSDL-S або ASWSDL.

WSDL-S представлення сервісу, що описаний у прикладі, може мати наступний вигляд:

```
<?xml version="1.0" encoding="iso-
8859-1"?>
      <definitions name="
      flightTicket"
      targetNamespace=http://lsdis.cs
      .uga.edu/projects/meteor-
      s/wsdl-
      s/examples/flightTicket.wsdl
      xmlns="http://www.w3.org/2004/0
      8/wsdl"
      xmlns:tns=http://lsdis.cs.uga.e
      du/projects/meteor-s/wsdl-
      s/examples/flightTicket.wsdl
      xmlns:xs="http://www.w3.org/200
      1/XMLSchema"
      xmlns:xsd1="http://lsdis.cs.uga
      .edu/projects/wsdl-
      s/examples/flightTicket.wsdl"

      xmlns:wssem="http://lsdis.cs.ug
      a.edu/projects/wsdl-
      s/examples/flightTicket.wsdl"
      xmlns:Flight="http://lsdis.cs.u
      ga.edu/projects/wsdl-
      s/ontologies/POOntology">
      <--опис типів -->
      <types>
      <xs:import
      namespace=http://lsdis.cs.uga.e
      du/projects/wsdl-
      s/examples/flightTicket.wsdl
      schemaLocation="http://lsdis.cs
      .uga.edu/projects/wsdl-
      s/examples/WSSemantics.xsd"/>
      <xs:schema
      xmlns:xs="http://www.w3.org/200
      1/XMLSchema"
      targetNamespace=http://lsdis.cs
      .uga.edu/projects/wsdl-
      s/examples/flightTicket.wsdl
      xmlns="http://lsdis.cs.uga.edu/
      projects/wsdl-
      s/examples/flightTicket.wsdl">
      <!--Semantic annotation is
      added directly to leaf element
      -->
      <xs:complexType
      name="processflightTicketReques
      t">
                <xs:all>
```

```
            <xs:element
            name="locationToInf
            o" type="xs:string"
            wssem:modelReferenc
            e="POOntology#Locat
            ion />
            <xs:element
name="locationFromInfo"
type="xsd1:string"/>
            <xs:element
            name="dateitem"
            type="xs:date"
            wssem:modelReferenc
            e="POOntology#Date
            />
            <xs:element
            name="ClientInfo"
            type="xs:string"
            wssem:modelReferenc
            e="POOntology#Clien
            t />
          </xs:all>
  </xs:complexType>
<!--Semantic annotation is
added directly to leaf element
-->
        <xs:element
name="processflightTicketRespon
se" type="xs:string"

wssem:modelReference="POOntolog
y#Flight"/>
        </xs:schema>
</types>
<interface
name="PurchaseOrder">
<operation
name="processflightTicket"
pattern="wsdl:in-out">
             <input
messageLabel
="processflightTicketRequest"
element="tns:processflightTicke
tRequest"/>
             <output
messageLabel
="processflightTicketResponse"
             element="
processflightTicketResponse"/>
           <!--Precondition and
effect are added as extensible
elements on an operation-->
           <wssem:precondition
name="ExistingLocationsPrecond"

wssem:modelReference="POOntolog
y#LocationsExist"/>
           <wssem:effect
name="FlightBookedEffect"

wssem:modelReference="POOntolog
y#flightBooked"/>
           </operation>
   </interface>
</definitions>
```

У WSDL-S, як і у більшості формальних мов, зв'язок з онтологією домена *POOntology* здійснюється за допомогою використання простору імен. Використання конкретних концептів онтологій у семантичних описах вхідних та вихідних параметрів, перед- та пост-умов досягається використанням префіксу з ім'ям онтології перед назвою концепта. Наприклад, конструкція *<xs:element name="locationToInfo" type="xs:string" wssem:modelReference= "POOntology#Location />* вказує, що елемент *locationToInfo* є екземпляром концепта *Location* онтології *POOntology*. Зв'язок онтологій між собою, наприклад, онтології конкретного сервісу та онтології сервісу верхнього рівня, або (та) онтології домену прикладної області, досягається шляхом інтеграції онтологій. Побудова інтегрованої онтології може здійснюватися за допомогою застосування різноманітних операцій маніпуляції онтологіями, таких як відображення, узгодження, уточнення, пого-

дження онтології та інше, а також шляхом застосування до онтологій операцій алгебри онтологій. Ці питання були досліджені та викладені у [7].

Вирішення задачі автоматизованого виявлення семантичних Веб-сервісів

Загальні вимоги до алгоритму співставлення. Щоб знайти Веб-сервіси для виконання конкретних задач в бізнес-процесі, запитувач має, перш за все, визначити умови, які повинен задовільняти цей Веб-сервіс. Такий підхід називається співставлення на основі цілі (goal based matchmaking) [8, 9] та має справу з визначенням умов на оголошення Веб-сервісу та перевіркою, чи може Веб-сервіс задовільняти ці умови. Цілі, як правило, слідують з бізнес-цілей, та можуть виводитись з них автоматично або не автоматично.

Опис чи оголошення сервісу відповідає запиту, якщо запит є досить близьким до сервісу, що запитується. Тут необхідно специфікувати, що означає «досить близький». У найсуворішій інтерпретації, оголошення та запит є «досить близькими», якщо вони описують точно один і той самий сервіс. Але це визначення є занадто обмеженим. Потрібно більш гнучке визначення поняття «достатньої близькості». Тобто, необхідні механізми співставлення, які розпізнають ступінь подібності між оголошеннями сервісів та запитами. А в того, хто запитує сервіс, має бути можливість визначати ступінь гнучкості, яку вони надають системі. Чим менша гнучкість, тим менша вірогідність знаходження сервісів, що задовільняють вимогам.

Таким чином:

- механізм співставлення має підтримувати гнучке семантичне співставлення між оголошеннями сервісів та запитами на основі онтологій, що доступні сервісам та механізму співставлення;
- запитувач має володіти деяким контролем ступеня гнучкості співставлення, що дозволяється системі;
- механізм співставлення має заохочувати обидві сторони бути чесними у своїх описах у питаннях вартості;
- процес співставлення має бути ефективним: він не повинен навантажувати запитувача надмірними затримками, які перешкоджатимуть його ефективності.

Алгоритми співставлення взагалі є ключовим питанням в області досліджень Веб-сервісів, та є основою вирішення не лише задачі виявлення сервісів, але й їх автоматизованої композиції. У загальному випадку, співставлення (matchmaking) Веб-сервісів містить у собі співставлення IOPE описів (співставлення за входами, виходами, передумовами, та ефектами). Дійсно, входи, виходи, передумови та ефекти утворюють опис функціональних можливостей сервісу. Наочно, IO надають синтаксичну інформацію про Веб-сервіси, тоді як PE – відображають їх семантику. Метод, який використовується у співставленні входів-виходів відрізняється від того, що використовується для передумов та ефектів. Та семантики, які відображаються входами-виходами й передумовами та ефектами також різні. На сьогодні досягнуто значних результатів у співставленні Веб-сервісів за входами-виходами, але відсутній ефективний підхід щодо співставлення перед-умов та ефектів. У роботі [10] наводиться алгоритм PE спіставлення, на основі формалізма ДЛ SHOIN+(D).

Безперечна доцільність використання апарату ДЛ для вирішення багатьох задач Веб-сервісів обумовлена тим, що системи ДЛ забезпечують користувачів різними механізмами виводу, які виводять неявні знання з тих, що явно представлені, та більше того, на сьогодні існує вже чимало реалізованих механізмів міркувань (резонерів) ДЛ, що готові до використання. Але, варто пам’ятати, що окремою складною задачею залишається вибір такої ДЛ, що є компромисним рішенням між її виразністю та розв’язуваністю.

Далі пропонується алгоритм виявлення Веб-сервісів на основі ДЛ, який адоптує множину стратегій. Задача виявлення полягає у знаходженні всіх Веб-сервісів, що задовільняють запиту. Іншими словами, це можна назвати фільтрацією множини сервісів у репозиторії за запитом. Тобто оголошення кожного сервісу співставляють із запитом. При чому запит та-

кож є своєрідним оголошенням сервісу, а саме, того сервісу, який реалізує потрібну користувачеві функціональність. Таким чином, відношення між запитом та оголошенням сервісу ідентичні відношенням між оголошеннями двох сервісів.

Критерій та ступені відповідності

Спочатку розглянемо задачу виявлення для сервісів, що задаються лише входами-виходами. Основу вирішення такої задачі становить встановлення IO-відповідності (Input-Output).

Вважаємо, що оголошення сервісу відповідає запиту, якщо всі виходи запиту співпадають з виходами оголошення, а всі входи оголошення співпадають зі входами запиту. Точне співпадання знайти досить важко, тому визначають ступені відповідності і обирають сервіси з найбільшим ступенем відповідності. Базовий критерій виявлення полягає у тому, що сервіс має задовільняти потреби запитувача, а запитувач має забезпечувати сервісу всі входи, які необхідні для його коректного функціонування.

Критерій відповідності. Якщо $\mathcal{T}$ – ациклічний Tbox ДЛ $\mathcal{L}$, $S = (In_s, Out_s)$ – сервіс та $Q = (In_q, Out_q)$ – запит, де:

- In_q – кінцева множина входів запиту,
- In_s – кінцева множина входів оголошення сервісу,
- Out_q – кінцева множина виходів запиту,
- Out_s – кінцева множина виходів оголошення сервісу, що виражені твердженнями $\mathcal{L}$, то відповідність сервісу запиту гарантується включеннями $In_s \subseteq In_q$ та $Out_q \subseteq Out_s$.

Таким чином, сервіс відповідає запиту, якщо запит містить всі вхідні твердження сервісу та, можливо, ще додаткові, а множина вихідних тверджень запиту, навпаки, має бути вужча за множину вихідних тверджень сервісу, тобто сервіс може повертати більше ефектів ніж того потребує запит.

Запити співставляються за цим критерієм з усіма оголошеннями сервісів, які є у реєстрі. Як тільки знайдена відповідність запиту та оголошення, вона записується, щоб знайти відповідність більш високого ступеня. Ступінь успіху залежить від його виявленого співпадання. Ступінь відповідності між двома входами або двома виходами залежить від відношення між концептами, які пов'язані з цими входами або виходами, а саме, мінімальною відстаню між цими поняттями у дереві таксономії. Розрізняють такі ступені відповідності [11, 12]:

- *Exact* – точне співпадання:
  - $Out_q = Out_s$ – еквівалентність, або
  - $Out_q \; subclassOf \; Out_s$ – результат точний у припущенні, що оголошуючи Out_s провайдер сервісу повинен забезпечити виходи, що погоджені з кожним безпосереднім підтипом Out_s.
- *Plug In* – більш слабкий зв'язок $(Out_s \; subsumes Out_q)$. Якщо Out_s $subsumes \; Out_q$, то Out_s – це множина, яка включає виходи Out_q, або іншими словами, Out_s може бути підключений замість Out_q.
- *Subsumes* $(Out_q \; subsumes Out_s)$ – якщо Out_q $subsumes$ Out_s, то провайдер не повністю задовільняє запит. Це означає, що запитувач може використовувати провайдера для досягнення своєї мети, але йомовірно йому прийдеться модифікувати свій план або виконати інші запити, щоб завершити свою задачу.
- *Fail* – невдача, не знайдено будь-якого *subsumption* між Out_q та Out_s.

Ступінь відповідності відображається на дискретній шкалі, де найбільш переважним є точна відповідність (*Exact*), наступний рівень – *Plug In*, так як вихід, що повертається, може бути використаний замість того, що очікує запитувач. *Subsumes* – третій рівень, так як вимоги запитувача виконуються лише частково: оголошений сервіс може забезпечити лише деякі конкретні варіанти того, що бажає запитувач.

Самий низький рівень *Fail* – не прийнятний результат. Ці значення дозволяють ранжувати знайдені сервіси за ступенем відповідності запиту. Обираються сервіси з найбільшим ступенем відповідності виходів. Співставлення входів використовується лише як допоміжна (вторинна) оцінка, щоб розмежувати еквівалентно оцінені виходи.

Функціональна (IOPE) модель сервісу у загальному випадку буде мати вигляд:

$$S = \{s | s = (CI;\ I;\ O;\ CO)\}, \qquad (1)$$

де CI – передумови сервісу, I – список вхідних параметрів, O – список вихідних параметрів та CO – після-умови.

Запит сервісу [13], в цьому випадку, розширюється та визначається як

$$Q = (CI';\ I';\ O';\ CO'), \qquad (2)$$

де CI' – передумови, I' – список вхідних параметрів, O' – список вихідних параметрів та CO' – після-умови. Всі ці елементи є параметрами сервісу, що запитується.

Валідні рішення щодо запиту, у даному випадку, мають задовільняти наступним умовам:

(i) вони мають виробляти хоча б один вихідний параметр запиту;

(ii) вони мають використовувати вхідні параметри тільки з наданого списку вхідних параметрів та задовільняти передумови запиту;

(iii) вони мають виробляти ефекти запиту.

Деякі рішення можуть бути занадто обмеженими, але вони розглядаються як дійсні, якщо задовільняють вимогам вхідних та вихідних параметрів, перед/після умов та ефектів.

Окрім IOPE атрибутів, співставлятися можуть і категорії сервісів, і додаткові параметри сервісів, як наприклад, не функціональні характеристики. В такому випадку, мова йде про багаторівневе співставлення. Значення ступеней відповідності на різних рівнях мають різну вагу. Результуюче значення є агрегацією багаторівневих зважених оцінок.

Запитувач може вирішити будь-які невідповідності шляхом вирішення додаткової задачі або шляхом запиту до реєстру, щоб знайти додаткових провайдерів.

Розглянемо задачу виявлення на прикладі бронювання авіаквітків. Тобто треба знайти сервіс бронювання квитків на літак, директорія сервісів містить сервіси S_1 та S_2 (таблиця далі).

Таблиця

Сервіс	Входи	Передумови	Виходи	Після-умови
S_1	*StartL, DestinationL, DateFrom, DateTo, CCNumber*	*Location Exist*	*status, Offer*	-
S_2	*StartL, Dt, DestinationL, CCNumber*	*CorrectCard*	*status, Offer, TrNumber*	-
Запит				-
Q	*Start, Dt, Destination, CCNumber*	*CorrectCard*		-

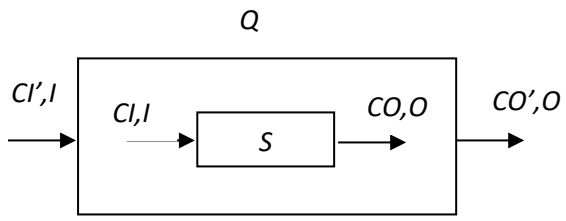
Сервіс S_1 має вхідні параметри *StartL*, *DestinationL*, *DateFrom*, *DateTo*, *CCNumber* та вихідні параметри *Status*, *Flight*. Для його виконання задана передумова *LocationExist*. Сервіс S_2 має вхідні параметри *StartL*, *DestinationL*, *Dt*, *CCNumber*, та вихідні параметри *Status*, *Flight*, *TrNumber* та для його виконання задана передумова *CorrectCard*. Для пошуку сервісу бронювання квитків заданий запит зі вхідними параметрами *Start*, *Destination*, *Date*, *CCNumber*, вихідними

параметрами *Status*, *Flight* та передумовою на номер кредитної картки *CorrectCard*.

Семантичні описи вхідних та вихідних параметрів сервісу мають бути такі самі як для параметрів запиту або мати відношення *subsumption*. Механізм виявлення повинен мати можливість вивести той факт, що параметри запиту *Start*, *Destination* та вхідні параметри *StartL*, *DestinationL* сервісів семантично одні й ті самі концепти. Це може бути виведено, використовуючи семантики онтології домена. (При цьому припускаємо, що сервіси та запит спираються на єдину інтегровану онтологію домену.) Запит також має передумову на *CCNumber*, яке повинно мати числове значення, що логічно має означати передумову сервісу, який шукається.

Визначення задачі виявлення. Заданий репозиторій $\mathcal{R}$ (3) та запит $\mathcal{Q}$ (4), задача виявлення може бути визначена як автоматичне знаходження множини сервісів S з репозиторію $\mathcal{R}$ таких, $s \in R$, та $I \sqsubseteq I'$, $A = A'$, $AO = AO'$, $O \sqsupseteq O'$}, та існує така інтерпретація $\mathcal{I}$, що $CI'^{\mathcal{I}} \subseteq CI^{\mathcal{I}}$, $CO^{\mathcal{I}} \subseteq CO'^{\mathcal{I}}$, де $\sqsubseteq$ – означає відношення включення (subsumes). Слід зазначити, що в даному випадку CI', CI, I, I', A, A', AO, AO', CO, CO', O, O' – це кінцеві множини вхідних/вихідних параметрів, перед- та пост-умов. Операції включення, що наведені в цьому визначенні, це операції на цих множинах.

Задача виявлення графічно пояснюється на рисунку.



Де $CI'^{\mathcal{I}} \subseteq CI^{\mathcal{I}}$, $CO^{\mathcal{I}} \subseteq CO'^{\mathcal{I}}$, $I' \supseteq I, O \supseteq O'$

Рисунок. Задача виявлення

Визначення сервісу та запиту за допомогою ДЛ-онтологій. ДЛ [14] дозволяють представити домен, що нас цікавить, у термінах концептів або описів (унарні предикати), що характеризують підмножини об'єктів (екземплярів) в домені, та ролей (бінарні предикати) на такому домені. Концепти визначаються виразами, які формуються за допомогою спеціальних конструкторів [14]: верхній концепт ($\top$), нижній концепт ($\bot$), кон'юнкція концептів ($\sqcap$), універсальний кваліфікатор ($\forall R.C$), числові обмеження ($\geq$ nR) та ($\leq$ nR) (для різних ДЛ цей набір конструкторів різниться).

Онтологія домену для наведеного прикладу була предствлена вище. Для опису наших сервісів, доповнимо TBox онтології *POOntology* декількома концептами та твердженнями. Додамо до онтології концепти: *Operation, TrackingNumber,* та визначимо наступні твердження, які дозволять нам описати параметри запиту та сервісів:

CreditCardNumber $\sqsubseteq$ *hasType.{String}*

Flight $\sqsubseteq$ *hasCreditCard.CreditCardNumber*

Operation $\sqsubseteq$ *hasTrackingNumber.Number*

Number $\sqsubseteq$ *hasType.Type*

Type $=$ *{String, Numeric, Boolean}*

Для вирішення задач Веб-сервісів визначимо онтологію сервісу верхнього рівня *ServiceOntology*.

Service, InputParameter, OutputParameter, PreCondition, PostCondition, Parameter, Name, Value, Type

Service $\sqsubseteq$ *has.InputParameter; Service* $\sqsubseteq$ *has.OutputParameter*

Service $\sqsubseteq$ *has.PreCondition; PreCondition* $\sqsubseteq$ *Condition*

PostCondition $\sqsubseteq$ *Condition; Condition* $\sqsubseteq$ *hasValue.Boolean*

Service $\sqsubseteq$ *has.PostCondition; InputParameter* $\sqsubseteq$ *Parameter*

OutputParameter $\sqsubseteq$ *Parameter; Parameter* $\sqsubseteq$ *has.Name*

Type $=$ *{String, Numeric, Boolean}*

$I_S \sqsubseteq InputParameter$

$O_S \sqsubseteq OutputParameter;$

$CI_S \sqsubseteq PreCondition;$

$CO_S \sqsubseteq PostCondition;$

$Parameter \sqsubseteq has.Value;$ $Name \sqsubseteq hasType.\{String\};$ $Value \sqsubseteq hasType.Type$

Запит є також сервісом, але абстрактним. Точніше, абстрактним описом сервісу, що реалізує поставлену бізнес-задачу. Таким чином, онтологію запиту Q можна визначити як:

TBox

$Q \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$Q \sqsubseteq =4has.InputParameter;$

$Q \sqsubseteq =2has.OutputParameter$

$Q \sqsubseteq =1has.PreCondition;$

$I_Q \sqsubseteq InputParameter$

$O_Q \sqsubseteq OutputParameter;$

$CI_Q \sqsubseteq PreCondition$

ABox

$Start: I_Q;$ $Destination : I_Q;$ $Dt : I_Q;$

$CCNumber : I_Q$

$Offer: O_Q,$ $status: O_Q;$ $CorrectCard : CI$

Start:Location; Destination:Location;

Dt : Date;

$Offer \sqsubseteq hasCreditCard.CCNumber;$

Offer:Flight;

status:Status

Сервіс S_1 можна визначити як:

TBox

$S_1 \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$S_1 \sqsubseteq = 5has.InputParameter;$

$S_1 \sqsubseteq =2has.OutputParameter;$

$S_1 \sqsubseteq = 1has.PreCondition;$

$I_{S1} \sqsubseteq InputParameter$

$O_{S1} \sqsubseteq OutputParameter;$

$CI_{S1} \sqsubseteq PreCondition$

ABox

$StartL: I_{S1},$

$DestinationL : I_{S1},$

$DateFrom : I_{S1},$

$DateTo: I_{S1},$

$CCNumber : I_{S1}$

$Status: O_{S1},$

$Offer: O_{S1};$

$EuropeanTrip: CI_{S1}$

StartL: Location зв'язок з онтологією *POOntology*

DestinationL: Location;

DateTo: Date;

DateFrom: Date

$Offer \sqsubseteq hasCreditCard.CCNumber;$

Offer: Flight;

status: Status

Сервіс S_2 можна визначити як:

TBox

$S_2 \sqsubseteq Service$ зв'язок з онтологією сервісу верхнього рівня

$S_2 \sqsubseteq = 4has.InputParameter;$

$S_2 \sqsubseteq =3has.OutputParameter;$

$S_2 \sqsubseteq = 1has.PreCondition;$

$I_{S2} \sqsubseteq InputParameter;$

$O_{S2} \sqsubseteq OutputParameter;$

$CI_{S2} \sqsubseteq PreCondition;$

$StartL: I_{S2},$

DestinationL: $I_{S2,}$

Dt: $I_{S2,}$

CCNumber: $I_{S2,}$

Offer: $O_{S2,}$

Status: $O_{S2.}$

TrNumber: $O_{S2\ ;}$

CorrectCard: CI_{S2}

StartL: Location зв'язок з онтологією *POOntology*

DestinationL: Location;

DateTo: Date;

DateFrom: Date;

Offer $\sqsubseteq$ *hasCreditCard.CCNumber;*

Offer:Flight;

status:Status;

TrNumber:TrackingNumber

Визначимо онтологію задачі виявлення *DiscoveryOntology*:

$Q \sqsubseteq Service;$

$S \sqsubseteq Service$

$ExactInput \equiv (I_Q \sqsubseteq I_S) \sqcap (I_S \sqsubseteq I_Q)$

$SubsumesInput \equiv (I_S \sqsubseteq I_Q)$

$SubsumedInput \equiv (I_Q \sqsubseteq I_S)$

$ExactOutput \equiv (O_Q \sqsubseteq O_S) \sqcap (O_S \sqsubseteq O_Q)$

$SubsumesOutput \equiv (O_Q \sqsubseteq O_S)$

$SubsumedOutput \equiv (O_S \sqsubseteq O_Q)$

$FailOutput \equiv \neg((OQ \sqsubseteq OS) \sqcap (OS \sqsubseteq OQ))$

$FailInput \equiv \neg((IQ \sqsubseteq IS) \sqcap (IS \sqsubseteq IQ))$

$ExactPreCondition \equiv (CI_Q \sqsubseteq CI_S) \sqcap (CI_s \sqsubseteq CI_Q)$

$SubsumesPreC \equiv (CI_s \sqsubseteq CI_Q)$

$ExactPostCondition \equiv (CO_Q \sqsubseteq CO_S) \sqcap (CO_s \sqsubseteq CO_Q)$

$SubsumesPostC \equiv (CO_Q \sqsubseteq CO_S)$

Можемо встановити наступні співвідношення між елементами кортежей описів сервісів та запиту.

Для сервісу S_2:

$I_{S2} \sqsubseteq I_Q$ – для входів,

$O_{S2} \sqsupseteq O_Q$ – для виходів,

$CI_Q \Rightarrow CI_{S2}$, в силу їх еквівалентності – для передумов.

Для сервісу S_1:

$I_{S1} \not\sqsubseteq I_Q$ – для входів,

$O_{S1} \sqsupseteq O_Q$ *та* $O_{S1} \sqsubseteq O_Q$ – для виходів.

Таким чином, сервіс S_2 задовільняє запиту, а сервіс S_1 – ні, тому що він вимагає як вхідний параметр *DateTo*, що не забезпечується запитом. Запит вимагає *Flight* та Status як вихідний параметр та S_2 виробляє *Flight*, *Status* та *TrNumber*. Додатковий вихідний параметр може просто ігноруватися. При такому підході до опису запиту та сервісів (за допомогою ДЛ), пошук, співставлення та висновок про відповідність сервісу та запиту може здійснюватися автоматично, за допомогою існуючих резонерів ДЛ.

Але слід зазначити, що при вирішенні задачі включення для пар концептів, що є елементами кортежу опису сервісу та запиту, наприклад, I_S та I_Q виконується не лише співставлення екземплярів, що задані іменами, аналізується їх семантика – зв'язок з онтолгією домена. Вирішення задачі знаходження найбільш специфічного класу для індивіда є стандартним алгоритмом ДЛ, як і задача включення на концептах.

Таким чином, якщо є два параметри *x* і *y* такі, що *x:C1* та *y:C2,* та $C1 \sqsubseteq C2$, то *x:C2*, та можна казати про відповідність вхідних параметрів.

Формально задачу виявлення можна визначити наступним чином:

$Q \sqsubseteq Service$

$S \sqsubseteq Service$

$$MatchingQueryXY(Q,S) \sqsubseteq$$

$$\exists x.Iq, \exists y.Is: (C1(x) \sqcap C2(y) \sqcap$$

$$((C1 \sqsubseteq C2) \sqcap (C2 \sqsubseteq C1)) \sqcup (C1 \sqsubseteq C2)).$$

Висновки

Наведений підхід базується на використанні ДЛ як для представлення предметної області задачі (онтологія домена), так і для вирішення безпосередньо задачі виявлення Веб-сервісів, шляхом представлення сервісів та запиту за допомогою онтологій.

Семантичні описи Веб-сервісу являють собою твердження або аксіоми ДЛ. Кожен сервіс представляється ДЛ-онтологією на основі загальної онтології сервісу верхнього рівня. Кожна онтологія конкретного сервісу базується на базі знань інтегрованої ДЛ-онтології домену.

Враховуючі, що опис профілей семантичних Веб-сервісів, що зберігаються в реєстрі, є WSDL документом, який доповнений семантичними анотаціями, тобто є звичайним XML, можливий автоматичний розбір цього XML-визначення та побудова онтології конкретного сервісу на основі онтології сервісу верхнього рівня.

Існуючі резонери ДЛ дозволять здійснювати автоматичне співставлення запиту та сервісів, сервісів один з одним за входами виходами. Це стандартні задачі виводу ДЛ. Відношення між концептами та індивідами, що представлятимуть вхідну та вихідну інформацію запиту, визначають ступінь відповідності сервісів, що співставляються та ранжувати їх за цим показником. Запропонований Алгоритм співставлення на основі ДЛ може бути застосований для співставлення сервісів при композиції.

Але, слід зазначити, що наведений підхід розглядає задачу встановлення відповідності окремого сервісу з репозиторію та запиту (пряме співставлення), але в реальності малоймовірно знайти один існуючий сервіс, що буде задовільняти запиту для вирішення задачі. Як правило, нам треба знайти множину сервісів, що разом дозволять нам визначити новий сервіс, який буде відповідати запиту, або іншими словами, «покривати» цей запит. Ця множина сервісів формує композитний сервіс, що покриває запит, та виконує поставлену бізнес-задачу (не пряме співставлення). Для формування такого сервісу може бути використана задача знаходження найкращого покриття запиту [15, 16], що з технічної точки зору належить до загальної структури для рерайтингу (перезапису) [17, 18], використовуючи термінологію з [19].

1. http://www.w3.org/2002/ws/
2. http://www.informit.com/articles/article.aspx?p=336265
3. Formal Description of Web Services for Expressive Matchmaking. Dipl.-Inform. Sudhir Agarwal, 2007 Karlsruhe
4. Web Service composition: Semantic Links based approach. Freddy L´ecu´, Doctor of Philosophy, 2008
5. Ortiz M., Calvanese D., and Eiter T. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics./AAAI, 2006.
6. Staab S., Studer R. Handbook on Ontologies. Second edition
7. Захарова О.В. Основні принципи побудови онтологічного граф-орієнтованого опису прикладної області. *Проблеми програмування.* 2010. № 4. С. 51–59.
8. Ruben Lara. Definition of semantics for web service discovery and composi tion. In Knowledge Web Deliverable D2.4.2, 2004.
9. D2.4.6 A Theoretical Integration of Web Service Discovery and Composition. Roberti Pierluigi (ITC-IRST) Marco Pistore (University of Trento) with contributions from: Walter Binder (EPFL), Ion Constantinescu (EPFL) Axel Polleres (UIBK), Holger Lausen (UIBK), Paolo Traverso (ITC-IRST), Michal Zaremba (NUIG). 2005. KWEB/2005/D2.4.6A/v1.0.
10. Hai Wang, Zengzhi Li. A Semantic Matchmaking Method of Web Services Based On SHOIN+(D). /Institute of Computer System Structure and Networks School of Electronics & Information Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049,

PR China hwang@mailst.xjtu.edu.cn, lzz@mail.xjtu.edu.cn.
11. Integrating Description Logics and Action Formalisms for Reasoning about Web-services. Franz Baader, Carsten Lutz, Maja Milieie, Ulrike Sattler, Frank Wolter. LTCS-Report 05-02.
12. Akkiraju R., et al. (2005, December 6). Web Service Semantics. WSDL-S. Available: http://www.w3.org/Submission/WSDL-S/.
13. http://life-prog.ru/view_zam2.php?id= 204&cat=5&page=13
14. Baader F. and Nutt W.; In Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. Basic Description Logics./ The Description Logic Handbook. P. 43–95. Cambridge University Press, 2003.
15. Baader F., Ku¨sters R., and Molitor R. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In T. Dean, editor, Proc. of the 16th Int. Joint Conf. on AI. P. 96–101. M.K, 1999.
16. Franz Baader, Ralf Kiisters, and Ralf Molitor LuFg Theoretische Informatik, RWTH Aachen. Computing Least Common Subsumers in Description Logics with Existential.
17. Beeri C., Levy A.Y., and Rousset M-C. Rewriting Queries Using Views in Description Logics. In L. Yuan, editor, Proc. of the ACM PODS , New York, USA. P. 99–108, Apr. 1997.
18. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
19. Franz Baader, Ralf Ku¨sters, and Ralf Molitor. Rewriting Concepts Using Terminologies. In Proc. of the Int. Conf.KRColorado, USA. P. 297–308, Apr. 2000.

References

1. http://www.w3.org/2002/ws/
2. http://www.informit.com/articles/article.aspx?p=336265.
3. Formal Description of Web Services for Expressive Matchmaking. Dipl.-Inform. Sudhir Agarwal, 2007 Karlsruhe.
4. Web Service composition: Semantic Links based approach. Freddy L´ecu´, Doctor of Philosophy, 2008.
5. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics./AAAI, 2006.
6. S. Staab, R. Studer. Handbook on Ontologies. Second edition.
7. Zakharova O. General Principes for building the ontological graph – oriented description of application area. /Problems of programming. - №4, 2010. pp.51-59.(Ukrainian).
8. Ruben Lara. Definition of semantics for web service discovery and composi tion. In Knowledge Web Deliverable D2.42, 2004.
9. D2.4.6 A Theoretical Integration of Web Service Discovery and Composition. Roberti Pierluigi (ITC-IRST) Marco Pistore (University of Trento) with contributions from: Walter Binder (EPFL), Ion Constantinescu (EPFL) Axel Polleres (UIBK), Holger Lausen (UIBK), Paolo Traverso (ITC-IRST), Michal Zaremba (NUIG). 2005. KWEB/2005/D2.4.6A/v1.0.
10. Hai Wang, Zengzhi Li. A Semantic Matchmaking Method of Web Services Based On SHOIN+(D). /Institute of Computer System Structure and Networks School of Electronics & Information Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, PR China hwang@mailst.xjtu.edu.cn, lzz@mail.xjtu.edu.cn.
11. Integrating Description Logics and Action Formalisms for Reasoning about Web-services. Franz Baader, Carsten Lutz, Maja Milieie, Ulrike Sattler, Frank Wolter. LTCS-Report 05-02.
12. R. Akkiraju, et al. (2005, December 6). Web Service Semantics. WSDL-S. Available: http://www.w3.org/Submission/WSDL-S/
13. http://life-prog.ru/view_zam2.php?id=204&cat=5&page=13
14. F. Baader and W. Nutt; In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. Basic Description Logics./ The Description Logic Handbook, pages 43–95. Cambridge University Press, 2003.
15. F. Baader, R. Ku¨sters, and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In T. Dean, editor, Proc. of the 16th Int. Joint Conf. on AI, pages 96–101. M.K, 1999.
16. Franz Baader, Ralf Kiisters, and Ralf Molitor LuFg Theoretische Informatik, RWTH Aachen. Computing Least Common Subsumers in Description Logics with Existential.

17. C. Beeri, A.Y. Levy, and M-C. Rousset. Rewriting Queries Using Views in Description Logics. In L. Yuan, editor, Proc. of the ACM PODS , New York, USA, pages 99–108, Apr. 1997.
18. Alon Y. Halevy. Answering queries using views: A survey. VLDB Journal, 10(4):270–294, 2001.
19. Franz Baader, Ralf Ku¨sters, and Ralf Molitor. Rewriting Concepts Using Terminologies. In Proc. of the Int. Conf.KRColorado, USA, pages 297–308, Apr. 2000.

Одержано 29.08.2017

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 25.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

Інститут програмних систем
НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.
E-mail: ozakharova68@gmail.com.