Вибір семантичного Веб-сервісу на рівні процесу: на прикладі eBay/Amazon/PayPal

(Переклад Ремарович С.)

Переклад статті «**SemanticWeb Service Selection at the Process-level: the eBay/Amazon/PayPal Case Study**» Ivan Di Pietro, Francesco Pagliarecci, Luca Spalazzi, Annapaola Marconi, Marco Pistore

Абстракт

Були запропоновані декілька підходів для вирішення вибору розподілених процесів, описаних як семантичні Веб-сервіси. Однак їх практична застосовність в реальних сценаріях композиції залишається відкритим питанням. Вирішення цієї проблеми вимагає, з одного боку, мати справу з сервісами, описаними як бізнес-процеси зі станами, і, з іншого боку, розглянути складні вимоги вибору, що стосуються і інтерфейсу сервісу, і його поведінки. Справді, у більшості існуючих підходів вибір здійснюється на основі "функціонального" опису сервісу, тобто з точки зору його входів, виходів, передумов і наслідків. У цій статті ми представляємо підхід до вибору сервісу на рівні процесу та оцінки його на реальній ситуації, що спричиняє високий рівень складності: eBay Веб-сервіси, Amazon E-Commerce сервіси та сервіс e-payment, запропонований PayPal. Підхід заснований на представленні сервісів на рівні процесу, тобто на BPEL і WSDL специфікаціях, і який розширює ці стандартні специфікації мінімальними семантичними анотаціями, які дозволяють виконувати ефективні, але все ж корисні, семантичні міркування для вибору Веб-сервісів на рівні процесу.

1 Вступ

Важливість опису Веб-сервісів на рівні процесу широко визнається. Розглянемо, наприклад, стандартну мову для опису бізнес-процесів, як BPEL [3], і найбільш популярні стандарти для семантичних Веб-сервісів, як OWL-S [8] і WSMO [1]. В описі на рівні процесу, Веб-сервіс не представлений у вигляді «атомарного» компонента - з його входами, виходами, передумовами і ефектами, які можуть бути виконані в одному кроці. Замість цього, інтерфейс сервісу описує його поведінку, тобто процес, який взаємодіє з іншими сервісами на різних стадіях, і які можуть мати різні керуючі конструкції, наприклад, послідовності, умови і ітерації. Ця інформація про модель процесу може бути використана для виконання складного різновиду автоматизованої перевірки, виявлення, вибору та композиції сервісів і для вираження вимог на рівні процесу [2]. Це особливо справедливо щодо композиції, про що свідчить [14, 11, 4, 23, 19]. Більшість підходів, які мають справу з описами складної поведінки, не мають справу з семантичною анотацією сервісів, і тому не можуть використовувати здатність виконувати міркування про те, що сервіси роблять. Це є блок засобів для композиції BPEL процесів [19], а також теоретичні основи для композиції сервісів, представлених у вигляді автоматів з кінцевим числом станів [11, 4]. З іншого боку, більша частина підходів, які були запропоновані не так давно, щоб використовувати семантику [14, 23, 1] можуть мати справу тільки з атомарними сервісами. Нечисленні винятки мають той недолік, що потребують всебічного семантичного опису процесів. Такі описи, засновані на виразних мовах, як OWL [13] або WSMO [1], забирають багато часу і сил, і є дуже важкими, щоб запропонувати їх на практиці для промислових застосувань.

Основна ідея нашого підходу полягає у відокремленні процедурного опису процесів від онтологічних описів і зв'язування їх через семантичні анотації. Основні характеристики підходу можна підсумувати таким чином:

• процедурна поведінка Веб-сервісу описана в BPEL [3]. Процес BPEL формально можна змоделювати як систему перехідного стану (State Transition System - STS) [23];

• обмін даними між процесами описується в стандартному файлі WSDL;

• семантика даних обміну описана в окремій онтологічній мові. Мова, яку ми використовуємо, - це WSML [1], що належить до сімейства дескриптивних логік [22];

• визначаємо мову анотації [18], що дозволяє нам зв'язати визначення даних (WSDL) і поведінки (BPEL) процесу з елементами онтології (WSML). Мова заснована на XML і, з теоретичної точки зору, вона належить до частини тверджень дескриптивної логіки. Такий підхід дозволяє нам анотувати тільки те, що нам потрібно, і залишити BPEL обов'язок опису поведінки;

• визначимо мову, яка може виражати вимоги на поведінку сервісу, який повинен бути перевірений, вибраний або скомпонований. Ця мова є темпоральною логікою на основі CTL (Computation Tree Logic – логіка дерев обчислень) [10], збагачена твердженнями концептів та ролей дескриптивної логіки;

• ми пропонуємо алгоритм граундінга, який включає в себе семантичні анотації в STS, що моделюють Веб-сервіси. Це дозволяє отримати модель STS, яку можуть обробляти існуючі контролери моделі (наприклад, [7]) і планувальники (наприклад, [5]), щоб вирішити проблеми перевірки, відбору і композиції.

У попередніх роботах був запропонований алгоритм виявлення [16] і алгоритм композиції [20], які використовують мінімалістську семантичну анотацію Веб-сервісів. У цій статті представлено комплексний підхід до вибору сервісу на рівні процесу та оцінка його на реальному сценарії: Веб-сервіси eBay, Amazon E-Commerce сервіси та сервіс e-payment, запропонований PayPal.

Робота побудована таким чином. У розділі 2 надається огляд підходу. У розділі 3 представлено Amazon-eBay-PayPal стандартний приклад і показані онтологія та анотації, що використовуються в цьому прикладі. Потім показано в подробицях різні фази підходу в роботі: у розділі 4 - як анотовані процеси BPEL транслюються в абстрактні STS, в розділі 5 - мова для визначення семантично анотованих вимог для вибору сервісу, перевірки і композиції, у розділі 6 наведено автоматизований вибір через перевірку моделі на нашому прикладі. Нарешті , у розділі 7 наведені деякі заключні зауваження.

2 Огляд підходу

Веб-сервіс можна охарактеризувати з точки зору його даних і його поведінки. Опис даних являє собою визначення типів даних, що використовуються в рамках сервісу, і це може бути зроблено за допомогою стандартної мови WSDL (Web Services Description Language). Цього недостатньо, оскільки WSDL представляє тільки статичну частину сервісу. З WSDL ми не знаємо фактичних потоків управління і даних у процесі: ми знаємо тільки інтерфейс Веб-сервісу і структури даних, які він використовує. Поведінкові аспекти сервісу можуть бути представлені на декількох мовах. Одна з найперспективніших мов, яка стане стандартом де-факто в поданні процесу, є BPEL (Business Process Execution Language). Як наслідок, ми будемо звертатися до BPEL в цій статті. WSDL плюс BPEL є «класичним», чисто синтаксичним поданням процесу. Збагачення цих описів семантичними анотаціями дозволяє використовувати автоматизовані методи міркувань, які допомагають вирішити ряд проблем, пов'язаних з сервісами в поширеному обчислювальному середовищі, такі як вибір, виявлення і композиція.

Підхід може бути описаний в чотири етапи: анотація, модель конвертування, заземлення, перевірка моделі.

Анотація: це фаза, в якій визначення даних і поведінки процесу збагачені посиланнями на онтологію. Онтологія повинна бути загальноприйнятою формалізацією деякої області. Важко мати таку онтологію, справді кожна організація може мати свою власну. Відповідність онтології є паралельною проблемою і ми не будемо заглиблюватися в це, тому що ми припускаємо, що у нас є загальна спільна онтологія для нашого домену. Існують різні підходи до анотації процесу (наприклад, SAWSDL [25]). Ми пропонуємо один підхід, який спрямований головним чином на збереження оригінального синтаксису BPEL і WSDL файлів. Таким чином, в запропонованому підході анотація поміщається в інший файл з посиланнями на BPEL і WSDL через вирази XPath.

Модель конвертування (translation): вона полягає у вираженні нашого процесу в іншій формі, яка може бути перевірена легко і автоматично. Зокрема, оскільки контролери моделі (model checker) зазвичай мають справу з деяким видом систем переходів станів, ми переводимо

анотований BPEL процес в анотовану систему переходів станів (*Annotated State Transition Systems* - ASTS). Цей крок може виконуватися автоматично.

Заземлення (grounding): це процедура, за допомогою якої семантичні анотації "звужені" до чисто синтаксичної форми, грубо кажучи, твердження концептів і ролі перетворюються в логічні висловлення. З технічної точки зору, кожен анотований стан - це моделювання бази знань тверджень, які виконуються в цьому стані. Це гарантує, що наші анотації можна трактувати за допомогою існуючих контролерів моделі, які працюють тільки з висловлюваннями (proposition). Заземлення застосовується як до анотованої STS, так і до специфікації мети, яка виражається в анотованій CTL. Після виконання алгоритму заземлення отримуємо заземлену (пропозиціональну) STS і заземлену (пропозиціональну) CTL специфікацію.

Перевірка моделі: вона полягає в перевірці, чи специфікація заземленої CTL підтверджується заземленою STS (якщо ні, передбачено контр-приклад). У наших експериментах ми використовували NuSMV [7], відомий впроваджений контролер моделей (model checker).

3 Огляд прикладу

У нашому випадку дослідження ми посилаємося на два реальні сервіси двох найбільших сайтів електронної комерції: eBay і Amazon. Заради зручного читання представимо абстрактний BPEL прикладу графічно (див. рис. 1 і рис. 2) (http://leibniz.diiga.univpm.it/ spalazzi/SWS/eCommerceCS.zip – **не працює!**).

Обидва сервіси дозволяють користувачеві шукати об'єкт. Можливі кілька механізмів пошуку, але ми будемо розглядати пошук тільки на основі ключових слів. У випадку, якщо користувач виявив предмет, який він шукав, і він задоволений його ціною, він може перейти до фази перевірки. Звичайно eBay має механізм пропозиції ціни, але ми не будемо розглядати його, а зосередимося на взаємодії "купити зараз (buy now)" або перевірка на сайті Express. Процес перевірки має фазу платежу, яка може бути виконана різними способами. Найцікавішою частиною є фаза платежу, яка має різні потоки в двох сервісах. У Amazon ми припускаємо, що платіж (операція *payMe*) здійснюється тільки за допомогою PayPal. Замість цього, в eBay оплата може бути виконана двома гілками: на лівій оплата проводиться з PayPal , в той час як на правій це робиться за допомогою кредитної карти. Як ми можемо висловити той факт, що ми маємо два різних методи оплати і які ці методи? Ця інформація може бути необхідна для вибору або композиції цих сервісів. Чи може ця інформація бути отримана з типів даних? Взагалі кажучи, відповідь буде негативною. Визначення типу даних не допомагає з'ясувати, який конкретний спосіб оплати пропонується. У нашому прикладі це означає, що ті ж самі повідомлення можуть бути використані для обох методів оплати: PayPal та кредитної картки, створюючи неоднозначність для вибору. Використання семантичних анотацій, посилаючись на онтологію електронної комерції, дозволяє чітко визначити цей вид інформації.

За підходом [20, 16] для кожного Веб-сервісу ми маємо: (1) онтологію, яка визначає відповідну термінологію; (2) процес інтерфейсу, який визначає необхідні взаємодії, щоб виконати сервіс, і (3) його анотацію, яка визначає (частково) відповідності між онтологією і процесом. Онтологія прикладу (рис. 3) дуже проста, але вона виражає основні концепти, корисні щоб змоделювати домен електронної торгівлі. У визначенні *Checkout* та *Search* у нас є атрибути *result* і *searchResult* відповідно, значення яких обмежені приналежністю до концепту *Result*. Можливими значеннями (екземпляри) *Result* є *CustomCode*, *Failure*, *Success*, *Warning*, ..., як показано на рис. 3.

На другому кроці ми анотуємо значимі змінні і дії наших процесів. Ми пропускаємо анотацію файлу WSDL (анотація типів даних) і переходимо безпосередньо до анотації BPEL файлів (анотація поведінки). Припустимо, ми повинні сказати, що після активності *searchResponse* і *searchResult* (див. рис. 1 і рис. 2 відповідно), пошук був проведений з успішним результатом. Це означає додавання наступних тверджень концепту і ролі до тих видів діяльності:

search : Search search.Result = Success.

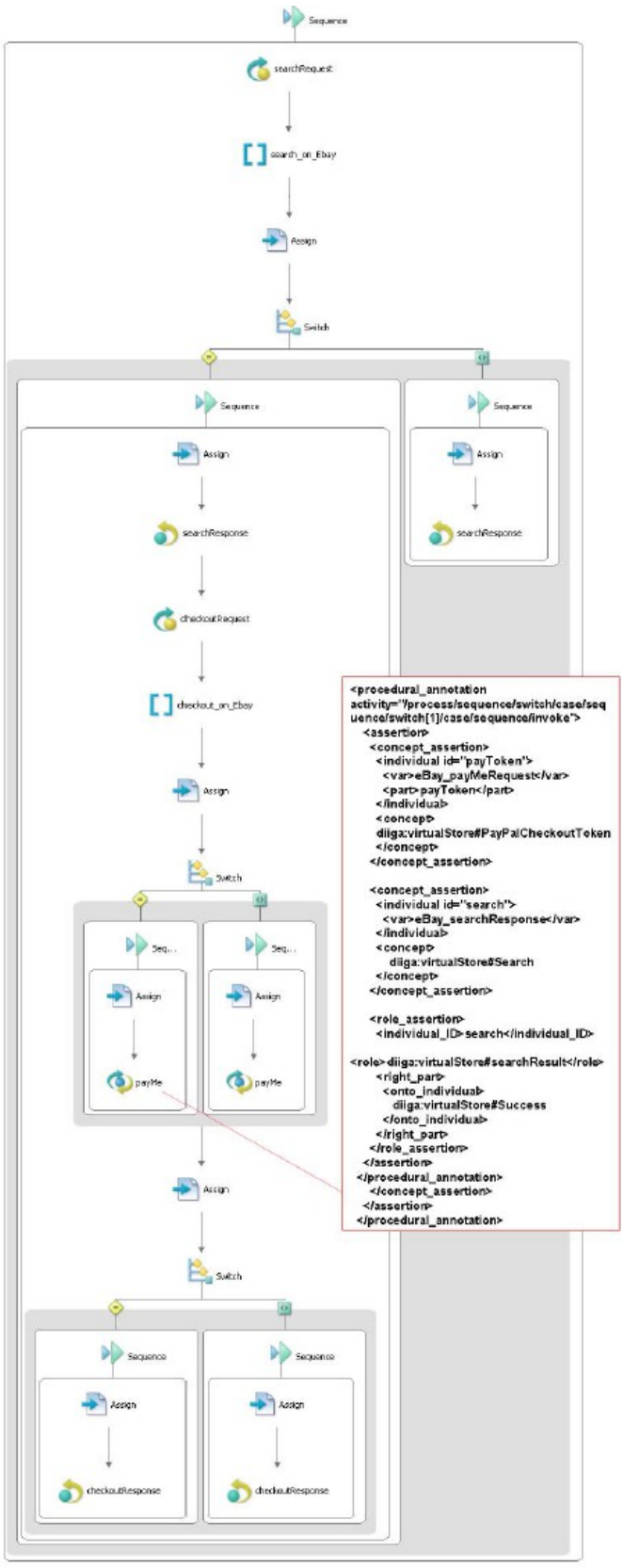

Рисунок 1 - BPEL для е-комерційного сервісу eBay

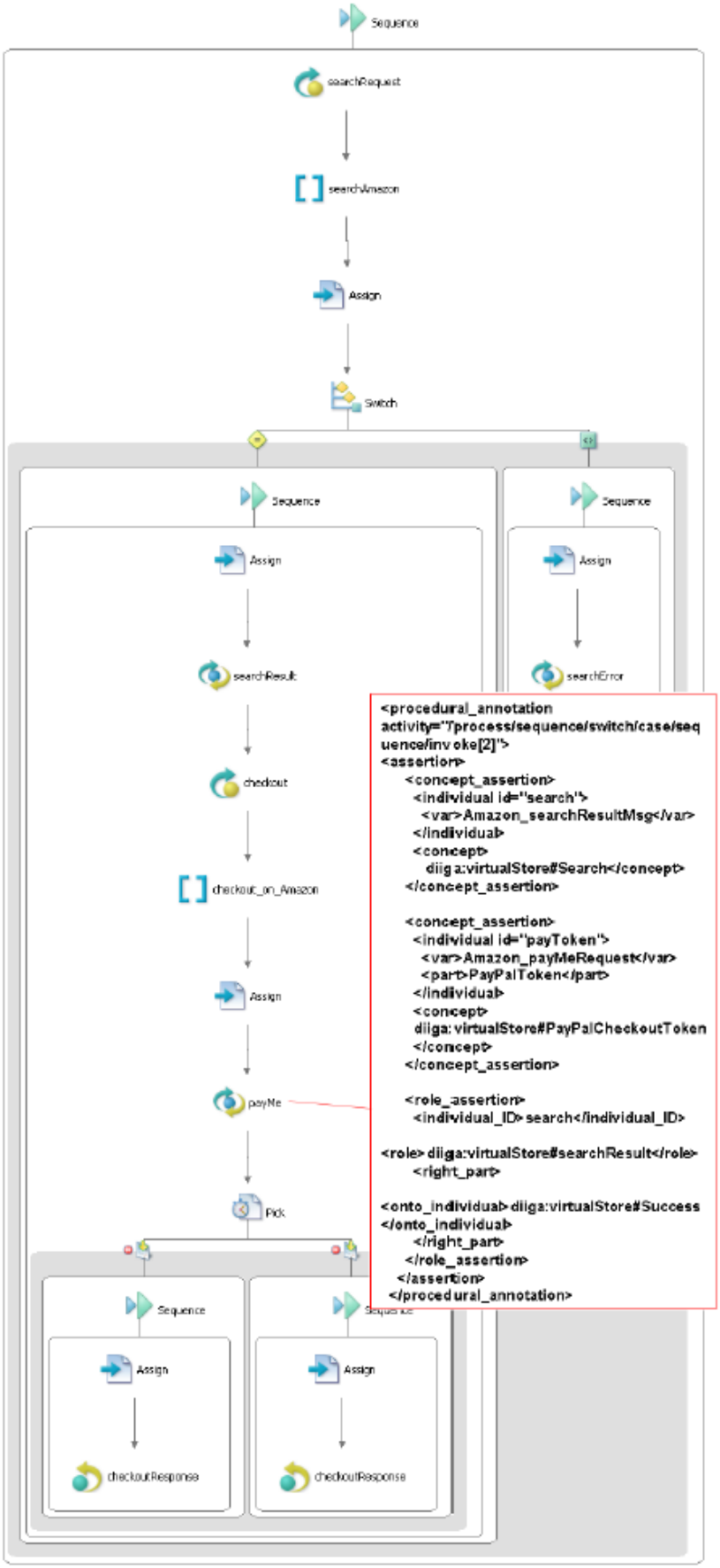

Рисунок 2 - BPEL для е-комерційного сервісу Amazon

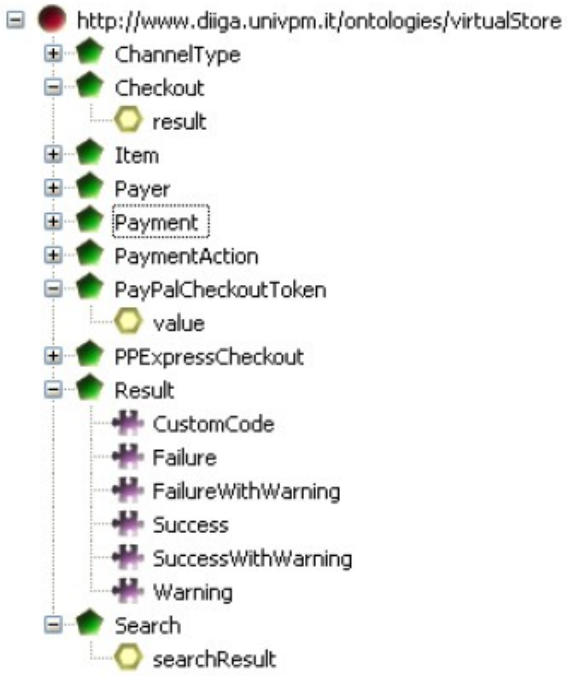

Рисунок 3 – WSML онтологія е-комерції

Точно так само і в іншій гілці BPEL процесів ми повинні анотувати невдачі. Для дій, названих *checkoutResponse*, у двох останніх гілок BPEL процесів анотації схожі, більше того, в лівій гілці маємо:

checkout : Checkout checkout.Result = Success,

тоді як в правій гілці маємо:

checkout : Checkout checkout.Result = Failure.

Важлива анотація має бути пов'язана з діями оплати, де використовується PayPal. Це визначається, помістивши твердження для концепту *PayPalCheckoutToken* в анотацію активності *payMe* в лівій гілці на рис. 1. Та ж анотація використовується для Amazon у її єдиній дії оплати (див. рис. 2).

4 BPEL процеси як анотовані STS

Ми кодуємо BPEL процеси (розширені семантичними анотаціями) як *анотовані системи перехідного стану* (*annotated state transition systems - STS*). STS описують динамічні системи, які можуть бути в одному з їх можливих станів (деякі з яких позначені як *вихідні стани*), і можуть розвиватися до нових станів в результаті виконання деяких *дій*. Ми розділяємо дії на *вхідні дії*, *вихідні дії* і τ. *Вхідні дії* являють собою прийом повідомлень, *вихідні дії* представляють повідомлення, передані до зовнішніх сервісів, а τ є спеціальною дією, називається *внутрішня дія*, яка представляє внутрішні еволюції, які не видно зовнішнім сервісам. Іншими словами, τ представляє той факт, що стан системи може розвиватися, не виробляючи будь-який вихід і без споживання будь-якого входу (це є наслідком того, ми використовуємо *абстрактний* BPEL, де внутрішні дії «непрозорі»). *Відношення переходу* (*transition relation*) описує, як стан може розвиватися на основі входів, виходів або внутрішньої дії τ. Ми припускаємо, що нескінченні петлі переходів τ не можуть з'явитися в системі. У анотованій STS ми пов'язуємо набір *тверджень концептів* та *тверджень ролі* з кожним станом. Це конфігурує стан як assertional компонент (або ABox) системи подання знань на основі даної дескриптивної логіки, де онтологія грає роль термінологічного компонента (або TBox). Таким чином, *твердження концепту* являють собою формули виду $a : C$ (або $C(a)$) і заявляють, що даний індивід a належить (інтерпретації) концепту C. *Твердження ролі* є формули виду $a.R = b$ (або $R(a, b)$) і заявляють, що даний індивід b є значення

ролі R для a. Як наслідок, кожну дію можна розглядати як перехід зі стану, що знаходиться в ABox, в інший стан, що міститься в іншому ABox. Що стосується циклів, мінімалістичний підхід дозволяє нам використовувати інваріанти до анотованих дій, які повинні бути повторені.

Визначення 1 Анотована система переходів (Annotated state transition system).

Анотована система переходів – це кортеж $\langle \Sigma, \mathsf{T}, \Lambda \rangle$ де:

$\Sigma = \langle S, S^0, \mathsf{I}, \mathsf{O}, R \rangle$ - це система переходів;

S – кінцевий набір станів;

$S^0 \subseteq S$ – набір початкових станів;

I – кінцевий набір вхідних дій;

O – кінцевий набір вихідних дій;

$R \subseteq S \times (\mathsf{I} \cup \mathsf{O} \cup \{\tau\}) \times S$ – відношення переходу;

T – термінологія анотації (TBox);

$\Lambda : S \rightarrow 2^{\mathsf{A}_\mathsf{T}}$ - функція анотації, де A_T - набір всіх тверджень концептів і тверджень ролей, визначених над T.

Представлення зазначених вище процесів як анотованих STS має термінологію T, яка задається WSML онтологією. Відповідний перелік концептів просто повідомляється в розділі CONCEPTS системи ASTS.

```
CONCEPTS
Search, PayPalCheckoutToken, Checkout, Result, PaymentAction, ChannelType
```

Функція Λ містить для кожного стану набір глобальних тверджень (включаючи твердження концептів, оголошених в онтології) і тверджень, якщо такі є, пов'язаних з цим станом. Як приклад ми наводимо функцію Λ тільки для одного анотованого стану, а саме стану, в якому виконується платіж за допомогою PayPal (див. рис. 1 і рис. 2).

```
ANNOTATION FUNCTION
LAMBDA(payMe˙Sync)
Failure : Result, Success : Result, Warning : Result,
SuccessWithWarning : Result, FailureWithWarning : Result,
CustomCode : Result, Sell : PaymentAction, Order : PaymentAction,
Authorization : PaymentAction, eVendorItem : ChannelType,
search : Search, search.searchResult = Success,
payToken : PayPalCheckoutToken
```

Останні три твердження були додані під час фази анотації, в той час як інші, прямо випливають із онтології і присутні в кожному стані, так як вони є глобальними.

5 Умови на анотованих STS

Для того, щоб виразити вимоги перевірки, вибору і композиції, ми повинні висловити умови на анотованій STS, тобто умови на твердження концептів і ролі, які виконуються в даних станах. Почнемо з поняття кон'юнктивного запиту в дескриптивній логіці, як визначено в [15].

Визначення 2 Кон'юнктивний запит (*Conjunctive Query*)

Кон'юнктивний запит q до $\langle \mathsf{T}, \Lambda(\mathrm{s}) \rangle$ - це набір атомів $\{p_1(\bar{x}_1), ..., p_n(\bar{x}_n)\}$, де кожен $p_i(\bar{x}_i)$ є або $p_i(x_i)$, або $p_i(x_{i,1}, x_{i,2})$ та $\bar{x}_i$ є кортеж змінних або індивідів:

$p_i(x_i) = x_i : C_i \quad p_i(x_{i,1}, x_{i,2}) = x_{i,1}.R_i = x_{i,2}$

$\mathsf{V}(q)$ позначає набір змінних q і $\mathsf{C}(q)$ позначає набір індивідів q. Тому $\mathsf{VC}(q) = \mathsf{V}(q) \cup \mathsf{C}(q)$

позначає набір змінних і індивідів q. Коли $V(q) = \varnothing$ у нас є заземлений кон'юнктивний запит, тобто кожен x_i , $x_{i,1}$ або $x_{i,2}$ є індивідами. Твердження концепту в пропозиціональній умові інтуїтивно позначає типову проблему дескриптивної логіки: *проблема пошуку виводу* (*retrieval inference problem*). Нехай $x : C$ буде твердженням концепту мети, тоді проблема пошуку виведення є проблемою знаходження для кожного стану s всіх індивідів, згаданих у ABox $\Lambda(s)$, які є екземплярами концепту C стосовно даного TBox T. Неоптимізований алгоритм пошуку може бути реалізований шляхом тестування для кожного індивіда, який з'являється в ABox, чи є він екземпляром концепту C. Після того, як ми отримаємо набір екземплярів $\{a\}$ для твердження концепту $x : C$, ми можемо замінити x у пропозиціональній умові на отримані екземпляри і переконатися, чи виконується ця умова. Тому *кон'юнктивний запит насправді позначає набір специфікацій*, які будуть перевірені замість однієї.

Темпоральна специфікація для анотованої STS є формулою CTL, що містить кон'юнктивні запити, як визначено в наступному:

Визначення 3 Темпоральна специфікація анотованої *STS*

Темпоральна специфікація $\varphi(q_1, ..., q_m)$ на $\langle \Sigma , T, \Lambda \rangle$ є формула, яка визначена на наборі кон'юнктивних запитів $\{q_1, ..., q_m\}$ наступним чином:

$\varphi = q_i \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg \varphi \mid AF\ \varphi \mid AG\ \varphi \mid EF\ \varphi \mid EG\ \varphi \mid AX\ \varphi \mid EX\ \varphi \mid$
$A(\varphi\ U\ \varphi) \mid E(\varphi\ U\ \varphi) \mid A(\varphi\ B\ \varphi) \mid E(\varphi\ B\ \varphi)$

Ми можемо розширити до темпоральних специфікацій визначення V наступним чином:

$V(\varphi(q_1, ..., q_m)) = V(q_1) \cup V(q_2) \cup ... V(q_m)$.

Визначення C і VC може бути розширено аналогічним чином. Заземлена темпоральна специфікація - це формула без змінних, тобто така, що $V(\varphi(q_1, ..., q_m)) = \varnothing$. Очевидно, що ми можемо анотувати інші темпоральні мови, як, наприклад, EAGLE [9], але для цілей цієї статті анотації CTL формул достатньо.

CTL є пропозиціональною темпоральною логікою розгалуженого часу (branching-time). Інтуїтивно, за нашим розширенням, темпоральна умова повинна бути перевірена на всіх можливих шляхах обчислення (послідовності станів), починаючи з поточного стану. У відношенні темпоральних операторів (*AF*, *EF*, *AX* і т. д.), вони зберігають те ж інтуїтивне значення, яке вони мають в стандартному CTL. Внаслідок того, що темпоральна специфікація має твердження концепту та ролі, *темпоральна умова позначає набір специфікацій* для перевірки замість однієї специфікації, а також для кон'юнктивного запиту.

Тепер, давайте припустимо, що ми хочемо перевірити деякі додаткові вимоги на сервіси, які ми знайшли. Ці вимоги можуть допомогти нам вибрати найбільш підходящий сервіс серед тих, які повертаються попереднім пошуком на основі ключових слів. Це так званий *вибір сервісу (service selection)*.

Вимога 1: я хочу знайти сервіс електронної комерції, який зможе виконати пошук, а потім, якщо деякі пункти повернуті, він гарантує можливість оплатити через мій PayPal рахунок.

У цій вимозі ми можемо знайти деяку відмінність від класичної пошукової системи. Дійсно, маємо процедурну специфікацію, яка розглядає оплату після того, як пошук був успішно виконаний. Очевидно, що нам знадобиться концепт часу або послідовність дій, щоб виразити поняття «після». По-друге, у нас є семантичний компонент, який дозволяє нам посилатися на рахунок PayPal, як описано у відповідній онтології електронної комерції. Використання семантики усуває можливе непорозуміння про використовувані терміни. Поки ми посилаємося на загальноприйняту онтологію, що ми маємо на увазі з пошуком, перевіркою і рахунком є очевидним. Ми можемо виразити нашу вимогу в анотованій CTL:

```
AF (!(x:Search & x.searchResult=Failure) ->
(x:Search & x.searchResult=Success & y:PayPalCheckoutToken))
```

6 Вибір анотованих STSs

Темпоральна специфікація може бути ефективно перевірена засобами перевірки моделей [6, 7]. Що стосується анотованих темпоральних специфікацій, то основна ідея полягає також у використанні перевірки моделі. Однак, традиційні контролери моделей не можуть бути використані, оскільки вони не можуть мати справу з онтологічними міркуваннями, які необхідні, щоб впоратися з анотаціями станів. Ми прийняли такий підхід, що дозволяє повторно використовувати існуючі контролери моделей (наприклад, NuSMV [7]), для того, щоб використовувати свої, дуже ефективні і оптимізовані методи перевірки. Цей підхід заснований на ідеї про те, щоб вирішити проблему дізнання, в яких станах твердження, що містяться в темпоральній специфікації, виконуються, до *завдання перевірки моделі* і алгоритм заснований на сервісі, який відповідає на запит (наприклад, алгоритм роботи [15]).

Формальний опис алгоритму заземлення можна знайти в [18]. Тут ми представляємо алгоритм в роботі через наш приклад. Ідея полягає в тому, щоб автоматично вибрати сервіс (між eBay і Amazon), який задовольняє вимогу 1. Виділимо кон'юнктивні запити, які становлять вимогу:

q_1 = {x:Search, x.searchResult=Failure}

q_2 = {x:Search, x.searchResult=Success, y:PayPalCheckoutToken}

Відповідний набір змінних і індивідів:

V($\mathcal{P}(q_1,q_2)$) = {search, payToken}

C($\mathcal{P}(q_1,q_2)$) = {Success, Failure}.

Якщо ми застосуємо алгоритм відповіді на запити до попередніх кон'юнктивних запитів і до нашої ASTS eBay, то отримаємо, в яких станах твердження, представлені в темпоральній умові, виконуються. Стан *searchResponse2* відповідає відмові пошуку (кон'юнктивний запит q_1) і стан *payMe_Sync* відповідає оплаті через PayPal (кон'юнктивний запит q_2). Відповідна CTL специфікація заземлення полягає в наступному:

```
AF (!(search:Search & search.searchResult=Failure) ->
(search:Search & search.searchResult=Success & payToken:PayPalCheckoutToken))
```

На даний момент, ми можемо перевірити здійснимість CTL специфікації для заземленого представлення процесів Amazon і eBay з використанням NuSMV контролера моделі. Як результат, ми маємо, що Amazon задовольняє специфікацію, в той час як eBay ні. Дійсно, eBay має альтернативну гілку оплати і контр-приклад, який охоплює цей шлях, надається.

7 Схожі роботи та висновки

Ця робота заснована і розширює роботи, повідомлені в [20, 16]. У цій статті представлено загальний підхід для вибору Веб- сервісу рівня процесу та оцінка його на прикладі реальної світу. Фреймворк WSMO [1] визнає важливість інтерфейсів, які описують поведінку сервісів, і пропонує використання медіаторів, щоб зіставити поведінку сервісів проти (виявлення) цілей. Тим не менш, WSMO структура включає представлення сервісів у своїй онтологічній моделі. Ми вирішили використовувати підхід знизу-вгору, не відмовлятися від існуючих і широко відомих технологій, таких як BPEL. Дійсно, BPEL дає нам поведінкові характеристики сервісу, тоді як у WSMO ми повинні були б виразити їх у властивостях оркестровки і хореографії. Роботи по WSDL-S і METEOR-S [21, 17, 24] забезпечують семантичну анотацію для WSDL. Він знаходиться близько по духу до нашого підходу, але не має справи з семантично анотованими описами Веб-сервісів (BPEL) рівня процесу. Робота [12] також близька за духом до нашої спільної мети подолання розриву між платформою Семантичного Веб і технологіями бізнес-процесу. Тим не менш, робота [12] зосереджена на проблемі розширення BPEL семантичною Веб-технологією для полегшення взаємодії Веб- сервісів, в той час як проблема автоматизованої композиції не вирішується. Останнім часом збільшується кількість робіт, що мають справу з проблемою композиції семантичних Веб-сервісів з урахуванням їх поведінкових описів [14, 26, 23, 1]. У цьому контексті дослідження дотримуються двох споріднених, але різних основних підходів: OWL-S [8] і WSMO [1]. Підходи на основі OWL-S [14, 26, 23] відрізняються від запропонованого у даній роботі, так як в OWL-S навіть процеси описуються як онтології, і, отже, немає ніякого способу відокремити

міркування про процеси і міркування про онтології. У підході, що здійснюється в WSMO, процеси представлені у вигляді абстрактних машин станів, добре відомий і загальний формалізм для представлення динамічної поведінки. Ідея, що лежить в основі WSMO, полягає в тому, що змінні абстрактних машин станів всі визначені з точки зору онтологічної мови WSMO. Наші процеси працюють замість цього на своїх власних змінних стану, деякі з них можуть бути відображені в окрему онтологічну мову, що дозволяє мінімалістський і практичний підхід до семантичних анотацій і ефективне міркування, щоб виявити, вибрати або виконати композицію сервісів автоматично. Справді, мета роботи по WSMO є створення спільної мови та механізму подання для семантичних Веб-сервісів, в той час як ми орієнтуємося на практичні проблеми забезпечення ефективних методів відбору та композиції семантичних Веб-сервісів автоматично. Було б цікаво дослідити, як наш підхід може бути застосований до WSMO абстрактних машин станів, а не BPEL процесів, і як ідея мінімалістських семантичних анотацій може бути розширена для роботи з рештою структури WSMO. Це завдання знаходиться в наших наукових дослідженнях. В роботі [2] автор пропонує поєднання SHIQ (D) DL і μ-числення. У нашому підході ми використовуємо CTL, щоб виразити темпоральні специфікації. CTL відноситься до категорії μ-обчислення, відповідно до мінімалістського характеру нашого підходу.

Література

[1] The Web Service Modeling Framework - http://www.wsmo.org/.
[2] S. Agarwal. A goal specification language for automated discovery and composition of web services. In *InternationalConference on Web Intelligence (WI '07)*, Silicon Valley, USA, NOV 2007.
[3] T. Andrews, F. Curbera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weeravarana. Business Process Execution Language for Web Services (version 1.1), 2003.
[4] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of E-Services that export their behaviour. In *Proc. ICSOC'03*, 2003.
[5] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. MBP: a Model Based Planner. In *IJCAI-2001workshop on Planning under Uncertainty and Incomplete Information*, 2001.
[6] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 1020 States and Beyond. *Information and Computation*, 98(2), June 1992.
[7] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: a new symbolic model checker. *InternationalJournal on Software Tools for Technology Transfer*, 2(4), 2000.
[8] T. O. S. Coalition. OWL-S: Semantic Markup for Web Services, 2003.
[9] U. Dal Lago, M. Pistore, and P. Traverso. Planning with a Language for Extended Goals. In *Proc. AAAI'02*, 2002.
[10] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 14, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, New York, N.Y., 1990.
[11] R. Hull, M. Benedikt, V. Christophides, and J. Su. EServices: A Look Behind the Curtain. In *Proc. PODS'03*, 2003.
[12] D. Mandell and S. McIlraith. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In *Proc. of 2nd International Semantic Web Conference (ISWC03)*, 2003.
[13] D. L. McGuinness and E. F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, 2004.
http://www.w3.org/TR/2004/REC-owl-features-20040210/.
[14] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proc. WWW'02*, 2002.
[15] M. Ortiz, D. Calvanese, and T. Eiter. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics. AAAI, 2006.

[16] F. Pagliarecci, M. Pistore, L. Spalazzi, and P. Traverso. Web service discovery at process-level based on semantic annotation. In *Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, Torre Canne, BR, Italy*, 2007.
[17] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEORS Web Service Annotation Framework. In *WWW04*, 2004.
[18] I. D. Pietro, F. Pagliarecci, and L. Spalazzi. Semantic Annotation of Web Services. Technical report, DIIGA — Universit`a Politecnica delle Marche, 2008.
http://leibniz.diiga.univpm.it/∼spalazzi/reports/TR-2008-01.pdf.
[19] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated Composition of Web Services by Planning at the Knowledge Level. In *Proc. IJCAI'05*, 2005.
[20] M. Pistore, L. Spalazzi, and P. Traverso. A minimalist approach to semantic annotations for web processes compositions. In *Proc. of the 3rd European Semantic Web Conference (ESWC 2006)*, Budva (Montenegro), 11–14 June, 2006. Springer–Verlag, Berlin, Germany.
[21] A. Sheth, K. Verna, J. Miller, and P. Rajasekaran. Enhacing Web Service Descriptions using WSDL-S. In *EclipseCon*, 2005.
[22] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.
[23] P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. In *Proc. ISWC'04*, 2004.
[24] K. Verma, A. Mocan, M. Zarembra, A. Sheth, and J. A. Miller. Linking Semantic Web Service Efforts: Integrationg WSMX and METEOR-S. In *Semantic and Dynamic Web Processes (SDWP)*, 2005.
[25] W3C Semantic Annotations for Web Service Description Language Working Group. Semantic Annotations for WSDL and XML Schema, 2007.
[26] D.Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S Web Services Composition using SHOP2. In *Proc. ISWC'03*, 2003.