CÁC PHƯƠNG PHÁP TÍCH HỢP CÁC ONTOLOGY VÀ CÁC QUY TẮC ĐỐI VỚI WEB NGỮ NGHĨA

Hoàng Nguyễn Tuấn Minh

Phòng Công tác học sinh, sinh viên, Trường Đại học Khoa học – Đại học Huế

Email: hntminh83@yahoo.com

TÓM TẮT

Trong tiến trình phát triển Web ngữ nghĩa, việc tích hợp các lớp khác nhau trong kiến trúc của nó đóng vai trò cốt lõi. Các quy tắc và ontology đóng vai trò rất quan trọng trong kiến trúc phân lớp của Web ngữ nghĩa, trong đó chúng được dùng để gán ý nghĩa và suy luận dữ liệu trên web. Việc tập trung vào nghiên cứu tích hợp lớp ontology với lớp các quy tắc là một hướng đang tập trung thu hút của rất nhiều nhà nghiên cứu trong những năm gần đây. Tuy đã có một số đề xuất được đưa ra để giải quyết vấn đề này song các giải pháp này vẫn còn các trở ngại khác nhau và chưa đạt hiệu quả như mong muốn. Bài viết này đưa ra một cái nhìn khá toàn diện về tổng quan và có sự so sánh, đánh giá các giải pháp như vậy trong việc kết hợp các quy tắc và ontology trong kiến trúc Web ngữ nghĩa.

Từ khóa: *Lập trình logic, Logic mô tả, Ontology, Web ngữ nghĩa.*

1. MỞ ĐẦU

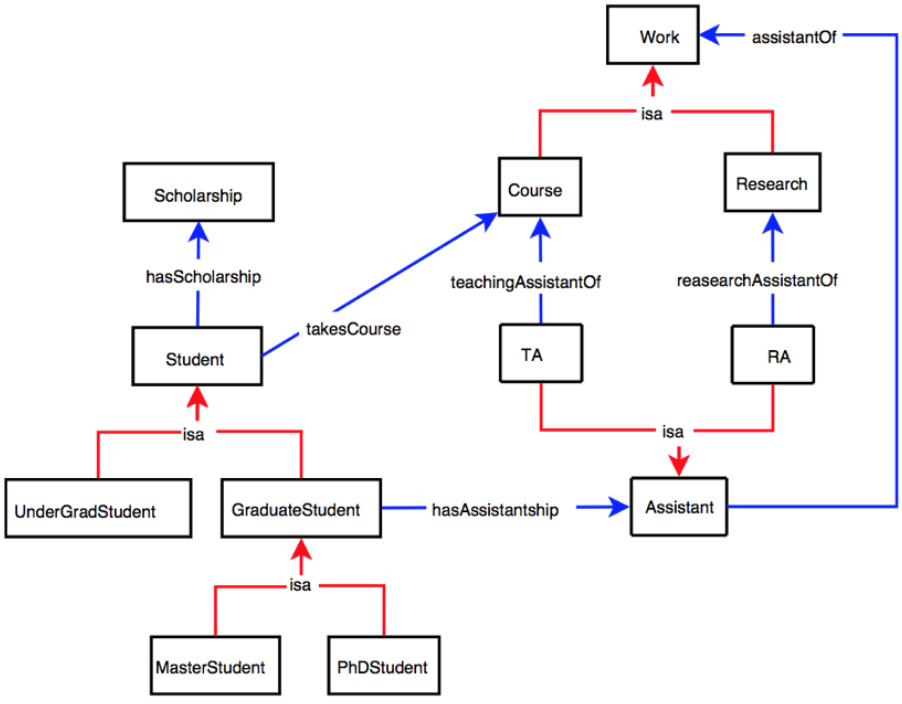
Web ngữ nghĩa ngày càng phát triển, một yêu cầu quan trọng của kiến trúc được phân lớp của web ngữ nghĩa là tích hợp các các quy tắc và các ontology trong đó chúng được sử dụng để gán ý nghĩa và suy luận dữ liệu trên Web. Trong khi lớp Ontology của Web ngữ nghĩa khá phát triển và ngôn ngữ Web Ontology (OWL) đã là một khuyến nghị của W3C từ một vài năm trước đây còn lớp các quy tắc ít phát triển hơn[1,2]. Việc tập trung vào nghiên cứu lớp quy tắc và tích hợp nó với lớp ontology là một hướng đang tập trung thu hút của rất nhiều nhà nghiên cứu trong những năm gần đây. Một số đề xuất đã được thực hiện để giải quyết vấn đề này nhưng không có một giải pháp đơn giản do trở ngại khác nhau và chưa đạt được hiệu như mong muốn. Trong bài viết, chúng ta xem xét vấn đề kỹ thuật cơ bản trong việc tích hợp các quy tắc và các ontology và phân loại các đề xuất theo các các phương pháp tiếp cận khác nhau cụ thể là các phương pháp tích hợp chặt chẽ ngữ nghĩa và các phương pháp tích hợp linh hoạt ngữ nghĩa đồng thời đưa ra các ví dụ tiêu biểu là $\mathcal{DL}$+*log* và chương trình logic mô tả cùng các đánh giá, so sánh giữa chúng và các hướng phát triển sau này.

2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1. Ontology

Thuật ngữ "Ontology" đã xuất hiện từ rất sớm. Trong cuốn sách Metaphysics của Aristotle đã định nghĩa: "*Ontology là một nhánh của triết học, liên quan đến sự tồn tại và bản chất các sự vật trong thực tế*". Các nhà nghiên cứu trong khoa học máy tính, đặc biệt là trong Trí tuệ nhân tạo (AI) mượn thuật ngữ này nhằm mục đích hỗ trợ việc chia sẻ và tái sử dụng kiến thức trong hệ thống AI. Cách tiếp cận này đã được Neches và các cộng sự đề xuất [3] *"Một ontology định nghĩa các thuật ngữ và các mối quan hệ cơ bản gồm từ vựng của một chủ đề cũng như các quy tắc kết hợp các thuật ngữ và mối quan hệ để định nghĩa các mở rộng cho từ vựng".* Theo định nghĩa này một ontology không chỉ bao gồm các thuật ngữ được định nghĩa một cách tường minh trong nó mà còn có tri thức có thể suy diễn được từ ontology. Vào năm 1998, Studer và các cộng sự đã đưa ra định nghĩa ontology khá phù hợp và chính xác hơn. *"Ontology là một đặc tả tường minh, mang tính hình thức của sự khái niệm hóa có thể chia sẽ được. Sự khái niệm hóa đề cập đến một mô hình trừu tượng của một số hiện tượng trong thế giới thực bằng cách xác định khái niệm liên quan đến hiện tượng đó. Tường minh có nghĩa là các khái niệm được sử dụng và các ràng buộc trên chúng được định nghĩa một cách rõ ràng. Hình thức đề cập đến máy có khả năng đọc và hiểu Ontology. Chia sẽ phản ánh quan điểm rằng một Ontology nắm bắt tri thức được chấp nhận bởi một cồng đồng."*Theo W3C một Ontology cung cấp một mô tả cho các phần tử sau: các lớp (hay các thực thể) trong một lĩnh vực xác định, các quan hệ giữa các lớp, các thuộc tính của các lớp.

Chúng ta xem xét ví dụ về ontology sinh viên như sau:



Hình 1. Ví dụ về ontology

Trong đó:

- Một *Student* có thể là một *UnderGradStudent* hoặc *GraduateStudent*;
- Một *GraduateStudent* có thể là *MasterStudent* hoặc *PhDStudent*;
- Một *Student* có thể có *Scholarship*;
- Một *Student* có thể tham gia *Course*(*s*);
- Work có thể là giảng dạy một *Course* hoặc làm *Reseach*;
- Một *Assistant* có thể là *TA* (*Teaching Assistant*) hoặc *RA*(*Reseach Assistant*);
- Một *Assistant* phải làm *Work*;
- Một *GraduateStudent* có thể có một *Assistant*;
- Một *TA* làm việc trên một *Course* và một *RA* làm việc *Reseach*

2.2. Quy tắc và chương trình logic trong Web ngữ nghĩa

2.2.1. Cú pháp

Cho $\Phi = (\mathcal{P}, \mathcal{C})$ là một bộ từ vựng ngôn ngữ bậc nhất với $\mathcal{C}$ là tập hữu hạn khác rỗng các hằng và $\mathcal{P}$ là tập các ký hiệu vị từ không chứa ký hiệu hàm. Cho $\mathcal{X}$ là tập các biến.

Một *hạng thức* là một biến từ $\mathcal{X}$ hoặc một ký hiệu hằng từ Φ. Một *nguyên tố* là một biểu thức có dạng $p(t_1, t_2,..., t_n)$ trong đó *p* là ký hiệu vị từ n ngôi, $n \geq 0$ từ Φ, và $t_1, t_2,..., t_n$ là các hạng thức. Một *literal l* là một nguyên tố *p* (*l* là literal dương) hoặc nguyên tố phủ định $\sim p$ (*l* là literal âm). *Phần bù* của *l* dương là $\sim p$ và của *l* âm là *p*. Một *literal phủ định ngầm (viết tắt NAF-literal)* là một literal *l* hoặc một literal phủ định mặc định *not l*.

Một quy tắc r là biểu thức có dạng : $a \leftarrow b_1, b_2,..., b_k, not\ b_{k+1},..., not\ b_m$ với $m \geq k \geq 0$ *(1)*

trong đó *a* là literal và $b_1,..., b_m$ là các literal hoặc các nguyên tố *đẳng thức (bất đẳng thức)* có dạng $t_1 = t_2$ $(t_1 \neq t_2)$ với t_1 và t_2 là các hạng thức. Literal *a* được gọi là *đầu* của quy tắc *r* và phép hội $b_1, b_2,..., b_k, not\ b_{k+1},..., not\ b_m$ là *thân* của quy tắc *r*, trong đó $b_1, b_2,..., b_k$ (hoặc, $not\ b_{k+1},...,not\ b_m$) là *thân dương (hoặc thân âm)*. Người ta dùng *H(r)* để ký hiệu literal *a* đầu của quy tắc, và *B(r)* để ký hiệu tập tất cả literal $B^+(r) \cup B^-(r)$ thân của quy tắc trong đó $B^+(r) = \{ b_1, b_2,..., b_k\}$ và $B^-(r) = \{ b_{k+1},..., b_m\}$. Nếu thân của quy tắc *r* rỗng (trong trường hợp $k = m = 0$) thì *r* là một *dữ kiện (Fact)*, chúng ta sẽ bỏ "$\leftarrow$" trong trường hợp này. Một *chương trình chính tắc P (hay đơn giản là chương trình P)* là một tập hợp hữu hạn các quy tắc. *P* là *một chương trình dương* nếu mọi quy tắc của nó đều không chứa phủ định "*not*".

2.2.2. Ngữ nghĩa

Một *vũ trụ Herbrand* của một chương trình *P*, ký hiệu HU_P, là một tập hợp tất cả các ký hiệu hằng xuất hiện trong *P*. Nếu không có ký hiệu hằng trong *P* thì $HU_P = \{c\}$, trong đó *c* là một ký hiệu hằng tùy ý trong Φ. Như thường lệ, các hạng thức, các nguyên tố, các literal, các quy tắc, các chương trình … là *nền* nếu và chỉ nếu chúng không chứa biến nào. Một *cơ sở Herbrand* của một chương trình *P* được ký hiệu là HB_P là tập tất cả các literal nền được xây

dựng từ các ký hiệu vị từ xuất hiện trong P và các ký hiệu hằng trong HU_P. Một *hiện hành nền* của một quy tắc $r \in P$ nhận được từ quy tắc r bằng cách thay thế mỗi biến trong r bằng một ký hiệu hằng trong trong HU_P, bằng cách sử dụng một phép thế θ cho mỗi biến trong r và loại bỏ tất cả nguyên tố đẳng thức và bất đẳng thức $t_1\theta = t_2\theta$ và $t_1\theta \neq t_2\theta$. Một hiện hành nền của một quy tắc r là *nhất quán* khi và chỉ khi nó không chứa các nguyên tố đẳng thức hoặc bất đẳng thức nào. Chúng ta ký hiệu *ground(P)* là một tập hợp tất cả các hiện hành nền nhất quán của các quy tắc trong P.

Một tập $X \subseteq HB_P$ của các literal *nhất quán* khi và chỉ khi $\{p, \neg p\} \not\subseteq X$ cho mọi nguyên tố $p \in HB_P$. Một *diễn dịch I* trong một chương trình P là một tập con nhất quán của HB_P. Một *mô hình* của một chương trình dương P là một diễn dịch $I \subseteq HB_P$ sao cho $B(r) \subseteq I$ dẫn đến $H(r) \subseteq I$ đối với mọi $r \in ground(P)$.

Phép chuyển đổi (hay phép chuyển đổi Gelfond-Lifschitz) của một chương trình P liên quan đến một diễn dịch $I \subseteq HB_P$ (ký hiệu là P^I) là một chương trình dương nền nó được nhận được từ *ground(P)* bằng cách xóa đi tất cả các quy tắc r mà $B^-(r) \cap I = \emptyset$ và xóa đi phần thân phủ định trong các quy tắc còn lại.

Một *tập trả lời* của một chương trình P là một diễn dịch $I \subseteq HB_P$ mà I là một tập trả lời của P^I. Tập hợp của tất cả các tập trả lời của một chương trình P được ký hiệu là *ans(P)*.

3. TÍCH HỢP CÁC QUY TẮC VÀ CÁC ONTOLOGY

Phần trên cung cấp cho chúng ta một cái nhìn tổng quan của ontology với mục đích hỗ trợ, chia sẻ và sử dụng lại các tri thức được biểu diễn hình thức trong hệ thống AI cũng như một phần cơ bản về quy tắc và chương trình logic trong Web ngữ nghĩa. Trong những năm qua lớp ontology đã đạt được thành công đến một mức độ nhất định với các khuyến nghị của W3C như RDF và OWL. Gần đây, các nhà nghiên cứu đã quan tâm đến việc tích hợp các lớp này và lớp các quy tắc với các mục đích sau:

- Nhìn từ quan điểm lập trình logic, chúng ta có thể sử dụng các ontology để cung cấp xác định các cá thể, các đối tượng từ các nguồn khác nhau để chia sẻ và tái sử dụng.
- Nhìn từ quan điểm ontology, ngôn ngữ quy tắc giống như các chương trình logic có khả năng khắc phục những trở ngại trong các hệ hình thức ontology dựa trên logic mô tả như khả năng diễn đạt mối quan hệ cao hơn với các vị từ nhiều ngôi, ràng buộc toàn vẹn và các trường hợp ngoại lệ khác.

Một số phương pháp được đưa ra để giải quyết vấn đề này song các giải pháp này vẫn còn các khó khăn khác nhau và chưa đạt hiệu quả như mong muốn. Tiếp theo, chúng ta sẽ đưa ra các vấn đề phát sinh khi đưa ra các phương pháp này và xem xét hai các tiếp cận trong việc tích hợp các quy tắc và các ontology cùng các đánh giá liên quan.

3.1. Các vấn đề tích hợp các quy tắc và các ontology

3.1.1. CWA so với OWA

Logic bậc nhất và các phân đoạn của logic mô tả áp dụng trên *giả thiết thế giới mở* (OWA), có nghĩa là "*nếu một mệnh đề không thể được suy ra từ những gì được biểu diễn trong một hệ thống, thì nó vẫn không thể được suy ra là sai.*". Sẽ là hợp lý khi xem xét Web ngữ nghĩa như hệ thống và như vậy áp dụng OWA lên nó. Trái lại, *giả thiết thế giới đóng* (CWA) [Reiter, 1978] giả định rằng "*những gì hiện đang không được biết đến là đúng được giả định là sai.*". Điều này không thực sự hợp lý trong cuộc sống thực. Trong lập trình logic, CWA liên quan chặt chẽ đến *phủ định khi thất bại* (NAF) mà nó nhận được *not p* từ sự thất bại trong việc nhận *p*. Để thấy rõ sự khác nhau giữa OWA và CWA chúng ta xem xét ví dụ sau :

Ví dụ 1 : *wine (X)* ← *whiteWine (X).*
nonWhite(X) ← not *whiteWine(X).*
wine(myDrink).

Theo cái không sẵn có *whiteWine(myDrink),* chương trình kết luận *nonWhite(myDrink),* trong khi một biểu diễn tương tự theo logic mô tả sẽ không chứng minh cho kết luận tương tự:

Ví dụ 2: *WhiteWine* ⊑ *Wine*
¬*WhiteWine* ⊑ *NonWhite*
myDrink ∈ *Wine*

Nguyên nhân cho cách xử lý này là theo OWA, không có đủ thông tin để đảm bảo một trong hai *myDrink* ∈ *WhiteWine* hoặc *myDrink* ∈ ¬ *WhiteWine*, do đó không có kết luận nào có thể tiếp tục được thực hiện.Mặc dù CWA không được áp dụng trong Web ngữ nghĩa vì ontology dựa trên logic mô tả nhưng nó vẫn cần thiết trong nhiều ứng dụng trong tích hợp thông tin.

3.1.2. Phủ định mạnh so với phủ truyền thống

Đôi khi, người đánh đồng phủ định mạnh với phủ định truyền thống, nhưng phủ định mạnh được sử dụng trong ASP trong thực tế hơi khác bản truyền thống của nó. Ví dụ sau đây cho thấy sự khác nhau này:

Ví dụ 3: *Wine(X)* ← *WhiteWine(X).* *WhiteWine* ⊑ *Wine.*
−*Wine(myDrink).* *myDrink* ∈ ¬*Wine.*

Trong khi trong cơ sở tri thức logic mô tả, chúng ta sẽ kết luận *myDrink* ∈ ¬ *WhiteWine* nhưng dữ kiện -*WhiteWine(myDrink)* không thể được chứng minh trong các thiết lập trình logic. Tuy nhiên, thêm một quy tắc *WhiteWine(X)* ∨-*WhiteWine(X)* trong ví dụ này sẽ giúp lấy được dữ kiện này.

3.1.3. UNA vs non-UNA

ASP hay hầu như tất cả các ngôn ngữ lập trình dựa trên logic hoạt động trên Giả thiết

tên duy nhất (UNA), mà về cơ bản có thể nói rằng hàm liên quan hằng số trên ngôn ngữ và các đối tượng trên các miền của diễn dịch là một song ánh. Giả định này không hợp lệ trong logic mô tả tổng quán. Các hàm ràng buộc hằng số và các đối tượng trong miền có thể không là đơn ánh cũng không là toàn ánh. Ví dụ, chương trình logic mô tả sau không có bất kỳ mô hình nào:

Ví dụ 4 : $\leftarrow friendOf\ (tweety,\ X),\ friendOf\ (tweety,\ Y),\ X \neq Y.$

$friendOf\ (tweety,\ joe\ ).$

$friendOf\ (tweety,\ pluto).$

Trong khi biểu diễn logic mô tả tương ứng :

$weety\ \in \leq 1 friendOf$

$friendOf\ (tweety,\ joe\ )$

$friendOf\ (tweety,\ pluto)$

có một mô hình trong đó *joe = pluto*. Sự khác biệt trong giả định cơ bản khi không được xem xét chắc chắn sẽ gây ra vấn đề khi chúng ta chú ý ngữ nghĩa trong sự tích hợp của hai hệ thống.

3.1.4. Khả năng quyết định

Chúng ta muốn lưu trữ một tích hợp có thuộc tính khả năng quyết định trong khi cho phép khả năng biểu diễn càng nhiều càng tốt. Lý tưởng nhất, chúng ta muốn có một hệ thống mà hệ thống đó cố gắng biểu diễn càng nhiều càng tốt theo các ràng buộc của một số lớp độ phức tạp của tính toán. Vấn đề phát sinh trong việc kết hợp hai hệ thống logic mô tả và lập trình dựa trên logic là khi chúng ta xem xét thực tế rằng hai hệ thống này sẽ cố gắng để tiếp cận và giải quyết vấn đề khả năng quyết định được từ hai góc độ khác nhau:

Khả năng quyết định trong ASP đạt được từ thực tế nó được dựa trên không có ký tự hàm, khi mà sự kế thừa thứ tự nền có thể được kiểm tra bằng sử dụng mô hình kiểm tra trong các tập con hữu hạn của các cơ sở Herbrand của chương trình. Nói cách khác, khả năng phụ thuộc vào sự hữu hạn của miền. Ngay cả ngữ nghĩa Prolog (sử dụng SLDNF), khi chúng ta xem xét các hàm trong ngôn ngữ kết hợp với đầy đủ đệ quy trái-phải có thể kết quả cũng không có khả năng quyết định được.

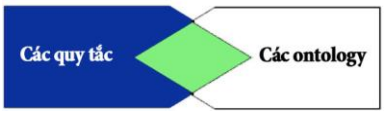
Trái lại, ngữ nghĩa logic mô tả duy trì khả năng quyết định bằng cách hạn chế các cấu trúc nó cung cấp để kết thúc trong một tập hợp con cụ thể của logic bậc nhất. Nhiệm vụ lý luận quyết định thành viên của lớp, xếp gộp, tính thỏa mãn … còn lại trên thực tế là chỉ có một số hữu hạn các cấu trúc cho phép trong các hệ thống thuật ngữ.

3.2. Các phương pháp tích hợp các quy tắc và các ontology

Có hai phương pháp phổ biến để kết hợp các quy tắc và các ontology phổ biến hiện nay là tích hợp chặt chẽ ngữ nghĩa và tích hợp linh hoạt ngữ nghĩa [5]. Bây giờ chúng ta sẽ phân tích các nguyên tắc của hai phương pháp này và đánh giá công việc liên quan đến chúng.

3.2.1. Tích hợp chặt chẽ ngữ nghĩa

Trong phương pháp này, các quy tắc được giới thiệu trực tiếp trong lớp ontology, tức là các tên khái niệm và vai trò có thể được sử dụng như là các tên vị từ trong các quy tắc. Cách tiếp cận như vậy có thể dễ dàng dẫn đến khả năng không quyết định được, ví dụ CARIN và SWRL. Mặt khác, DPL được đề xuất bởi Grosof (2003) bảo tồn khả năng quyết định rất hạn chế trong cú pháp của nó, do đó hạn chế khả năng biểu diễn. SWRL và DLP có thể được xem như hướng đi cho phép một số lượng lớn cho cách tiếp cận khác để phù hợp, chẳng hạn như $\mathcal{DL}$-log, các quy tắc DL-safe, r-hybrid KBs, và $\mathcal{DL}$+*log*. Những cách tiếp cận, để duy trì khả năng quyết định và mở rộng khả năng diễn đạt, đòi hỏi các ràng buộc điều kiện an toàn của các biến trong các quy tắc.[5]



Hình 2. Tích hợp chặt chẽ.

3.2.2. Tích hợp linh hoạt ngữ nghĩa

Trong phương pháp này, các quy tắc và các ontology (OWL/RDF) đóng vai trò trong các lĩnh vực khác nhau. Trong khi các quy tắc tập trung vào công việc lý luận thì OWL/RDF nhằm tăng mục đích của chúng trong vai trò là ngôn ngữ mô tả. Hai thành phần này không bị buộc bởi bất kỳ ràng buộc cú pháp nào, miễn là bên riêng mỗi chúng có khả năng quyết định, và sau đó chúng giao tiếp với nhau thông qua một "giao diện an toàn".

Nhìn từ quan điểm lớp các quy tắc, các ontology phục vụ như một nguồn thông tin mở rộng với một ngữ nghĩa độc lập có thể được cập nhật hoặc truy vấn thông qua một vị từ đặc biệt. Cách tiếp cận như vậy là chương trình logic mô tả [Eiter và cộng sự, 2005, 2007, Łukasiewicz, 2005, Eiter và cộng sự, 2008] và công cụ thực thi các quy tắc TRIPLE [SINTEK và Decker, 2002] chúng gọi các bộ suy luận logic mô tả bên ngoài.Phương pháp này thu hút sự quan tâm rất lớn của các nhà nghiên cứu trong những năm gần đây, trong nội dung của báo cáo chúng ta sẽ tập trung vào các vấn đề của việc tích hợp này bằng cách tiếp cận chương trình logic mô tả [6].

Hình 3. Tích hợp linh hoạt.

3.3. Các đại diện tiêu biểu của các phương pháp và các đánh giá

3.3.1. Tích hợp linh hoạt ngữ nghĩa: Chương trình logic mô tả

Chương trình logic mô tả mở rộng các chương trình tập trả lời (ASP) với các truy vấn đến cá cơ sở tri thức logic mô tả thông qua nguyên tử logic mô tả [8,9], trong đó các nguyên tử này có thể được điều chỉnh để cho phép truy vấn đến một cơ sở tri thức logic mô tả theo những cách khác nhau. Bằng cách như vậy các cơ sở tri thức logic mô tả và các chương trình logic được kết hợp dưới sự kiểm soát của các nhà thiết kế tri thức.

Trong các cài đặt thực tế kết hợp có sự tương tác được tách biệt rõ ràng giữa công cụ logic mô tả và một bộ xử lý ASP. Hai bên có thể chuyển giao kiến thức hai chiều thông qua nguyên tử logic mô tả mà chúng đóng vai trò như là một giao diện. Ý tưởng cơ bản của nguyên tử logic mô tả nhằm cung cấp một phương tiện để đặt ra các truy vấn đến cơ sở tri thức mô tả $\mathcal{T}$ từ chương trình *P*, bằng cách khai thác các truy vấn gốc của công cụ tri thức mô tả. Trong quá trình này, cũng kiến thức có thể chuyển từ *P* đến $\mathcal{T}$.

Chi tiết hơn, một truy vấn *Q* có thể là một thể hiện khái niệm/vai trò *C(X)/R(X,Y),* hoặc một bao hàm $C \sqsubseteq D$. Khi gửi một truy vấn, một nguyên tử logic mô tả cho phép sửa đổi các phần mở rộng (ABox) của $\mathcal{T}$, bằng cách thêm các khẳng định tích cực ($\uplus$) hay tiêu cực ($\cup$) được tính toán bằng các chương trình logic *P*. Các nguyên tử logic mô tả đánh giá đúng khi và chỉ khi $\mathcal{T}$ đã được sửa đổi chứng minh *Q*.

Ví dụ nguyên tử logic mô tả *DL*[*Wine*] ("*ChiantiClassico*") yêu cầu xem nó thỏa $\mathcal{T}$ |= *Wine* ("*ChiantiClassico*") không; một nguyên tử logic mô tả với một biến *DL*[*Wine*](*X*) đã đánh giá đúng cho tất cả các cá thể được biết đến *x* như là $\mathcal{T}$ | = *Wine*(*x*) thỏa.

Nguyên tử *DL*[*RedWine* $\uplus$ *my_red*; *Wine*](*X*) thêm tất cả các khẳng *RedWine*(*c*) vào $\mathcal{T}$, để *my_red* (*c*) thỏa chương trình logic *P*, trong khi *DL*[*RedWine* $\cup$ *my_white*; *hasColor*] (*X*, "*Red*") thêm tất cả các khẳng ¬*RedWine*(*c*) vào $\mathcal{T}$ để *my_white*(c) thỏa trong *P*.

Cụ thể hơn chương trình logic mô tả [8,9] là một cặp ($\mathcal{T}$, *P*) trong đó *P* gồm các quy tắc *r* có dạng $a_1, a_2,..., a_l \leftarrow b_1, b_2,..., b_k, not\ b_{k+1},..., not\ b_m$ *với* $l \geq 0$ *và* $k \geq m \geq 0$.

trong đó $a_1, a_2,..., a_l$ là các literal. Chúng ta gọi $a_1, a_2,..., a_l$ *là đầu của r* và phép hội $b_1, b_2,..., b_k, not\ b_{k+1},..., not\ b_m$ là *thân* của quy tắc *r*, trong đó $b_1, b_2,..., b_k$ (hoặc, $not\ b_{k+1},...,not\ b_m$) là *thân dương (hoặc thân âm)*. Người ta dùng *H(r)* để ký hiệu đầu của quy tắc, và *B(r)* để ký hiệu tập tất cả literal $B^+(r) \sqsubseteq B^-(r)$ thân của quy tắc trong đó $B^+(r) = \{ b_1, b_2,..., b_k\}$ và $B^-(r) = \{ b_{k+1},..., b_{nm}\}$. Một quy tắc không có đầu (l=0) là một *ràng buộc toàn vẹn. Một quy tắc r* trong phần đầu chỉ có một literal (l=1) là một *quy tắc bình thường.* Nếu thân của quy tắc *r* rỗng (trong trường hợp $n = m = 0$) thì *r* là một *dữ kiện (Fact),* chúng ta sẽ bỏ "¬ " trong trường hợp này.

Tập câu trả lời của một chương trình logic mô tả ($\mathcal{T}$, *P*) được xác định thông qua nền tảng tất cả các quy tắc trong P với một tập hợp các hằng *C*, trong đó *C* có chứa các hằng trong *P* và hằng số bổ sung từ $\mathcal{T}$. Một *mô hình* là một tập nhất quán của literal nền *M* được xây dựng từ các vị từ trong *P* và các hằng trong *C*. Một nguyên tử logic mô tả nền *DL*[<*Add*>; *Q*](**c**) đúng trong *M*, nếu và chỉ nếu $\mathcal{T} \cup <Add>^M \models Q(\mathbf{c})$. Lưu ý rằng $<Add>^M$ phụ thuộc vào *M*; điều này cho phép một dòng tri thức chạy từ *P* đến $\mathcal{T}$. Một mô hình *M* được gọi là một tập câu trả lời mạnh của ($\mathcal{T}$, *P*), nếu nó là mô hình nhỏ nhất của sP^M.

Chương trình logic mô tả quyết định được, với điều kiện đánh giá nguyên tử logic mô tả trên $\mathcal{T}$ quyết định được; Cụ thể hơn, nó có độ phức tạp là NEXP- đầy đủ với $\mathcal{T} \in \mathcal{SHIF}(\mathbf{D})$và $\mathbf{P}^{\text{NEXP}}$- đầy đủ với $\mathcal{T} \in \mathcal{SHOIN}(\mathbf{D})$ [8.9].

3.3.2. Tích hợp chặt chẽ ngữ nghĩa: $\mathcal{DL}$+*log*

$\mathcal{DL}$+*log* là phiên bản mới nhất trong các phần mở rộng của logic mô tả $\mathcal{ALC}$ với các quy tắc như $\mathcal{AL}$- *log*, r- và r^+ cơ sở tri thức lai. Các ngữ nghĩa chính của $\mathcal{DL}$+*log* có thể được tóm tắt như sau:

a, Có sự phân biệt giữa các vị từ quy tắc và các vị từ truyền thống.

b, miền vô hạn đếm được được cố định và các yếu tố *e* của nó có thể được truy cập trong tất cả các các diễn dịch với hằng số phân biệt c_e trong tương ứng một một; miền này được gọi là *giả thiết tên chuẩn* (UNA *Standard Names Assumption*).

c, Mô hình (còn được gọi là mô hình NM) của $\mathcal{KB}$= <$\mathcal{T}$,*P*> có dạng $\mathcal{I} \cup M$ trong đó $\mathcal{I}$ là một mô hình của các vị từ truyền thống, *M* là mô hình của các vị từ quy tắc, sau khi xóa các nguyên tử truyền thống được thỏa bởi $\mathcal{I}$ trong *P*.

d, Ngôn ngữ không có phủ định mạnh và phủ định yếu được giới hạn trong các vị từ quy tắc nhưng các vị từ truyền thống có thể xuất hiện trong các phần đầu của các quy tắc; những các ký hiệu hàm cũng không được xem xét.

e, Để đảm bảo tính quyết định, ngôn ngữ logic mô tả an toàn yếu (weak DL-safety) được sử dụng: mỗi biến *X* trong một quy tắc *r* phải xuất hiện trong một số nguyên tử thân dương của *r* và nguyên tử này phải có một vị từ quy tắc nếu *X* xuất hiện trong một nguyên tử với vị từ truyền thống trong phần đầu của *r*.

Lưu ý rằng ngôn ngữ logic an toàn yếu cho phép truy cập các cá thể không tên trong nguyên tử truyền thống. Ví dụ cho $\mathcal{KB}$= <$\mathcal{T}$, *P*> trong đó $\mathcal{T}$ = {*author* $\sqsubseteq$ *isAuthorOf*, *author*(*turing*)} và *P* bao gồm các quy tắc an toàn yếu:

$$scientist(X) \leftarrow isAuthorOf(X,Y), not\ likes(X, astrology);$$

Ở đây *isAuthorOf* là một vị từ truyền thống và *scientist* và *likes* là các vị từ quy tắc. Biến *Y* không xuất hiện trong bất kỳ nguyên tử với một vị từ nguyên tắc cũng có thể truy cập các cá thể không tên. Ta có $\mathcal{KB}$= <$\mathcal{T}$,*P*> $\models_{NM}$ *scientist* (*turing*), mặc dù *Y* có thể không được thể hiện và có thể thay đổi từ diễn dịch này đến diễn dịch khác. Các quy tắc tương tự thể hiện như một chương trình logic mô tả như sau:

$$scientist(X) \leftarrow DL[isAuthorOf](X,Y), not\ likes(X, astrology);$$

điều này không dẫn đến *scientist* (*turing*), Tuy nhiên, các chương trình logic mô tả được viết lại $scientist(X) \leftarrow DL[\exists isAuthorOf](X,Y)$, *not likes*(*X*, *astrology*) mang lại câu trả lời mong muốn; bằng cách sử dụng cú pháp mở rộng đề xuất trong [6, 7], các nguyên tử logic mô tả này có thể được thể hiện như là *DL*[*father*(*X*, *Y*)] (*X*).

$\mathcal{DL}$+*log* là quyết định được nếu phép hợp các truy vấn nối tiếp là quyết định được trong $\mathcal{T}$

3.3.3. Đánh giá

Một số tính năng đáng chú ý về ngữ nghĩa của chương trình logic mô tả và $\mathcal{DL}$*+log* được tóm tắt bảng sau:

Bảng 1. Bảng so sánh các tính năng giữa chương trình logic mô tả và $\mathcal{DL}$*+log*
(Trong đó: C: có, K: Không, CN: Đúng trong phần lớn các trường hợp.)

Tiêu chí	**Chương trình logic mô tả**	$\mathcal{DL}$***+log***
Phân biệt giữa vị từ và vị từ quy tắc	C	K
Miền vũ trụ cho P		
Vũ trụ Herbrand của P	K	CN
Ký hiệu kết hợp	C	CN
Các tên duy nhất trong vũ trụ Herbrand của P	C	C
Vị từ bình đẳng đặc biệt	CN	CN
Tri thức tích hợp: từ lý thuyết bậc nhất đến các quy tắc		
Mỗi mô hình đơn	K	K
Tính kế thừa	C	N
Tri thức tích hợp: từ các quy tắc đến lý thuyết bậc nhất		
Mỗi mô hình đơn	K	C
Tính kế thừa	C	K
Tính quyết định	CN	CN

Hàng đầu tiên xác định các hình thức nào có các từ vựng khác nhau đối với các tên vị từ truyền thống và vị từ quy tắc. Lưu ý rằng tính năng này không phải là đặc trưng của phương pháp tích hợp ngữ nghĩa linh hoạt, mặc dù nó có thể được xem như là một dấu hiệu cho thấy mức độ của các khớp nối giữa ngữ nghĩa lập trình truyền thống và lập trình logic.

Nhóm thứ hai của bảng xác định lựa chọn nào được thực hiện liên quan đến tập vũ trụ đối với các lập trình logic *P* của một cơ sở tri thức. Sự lựa chọn là khác nhau giữa dùng một miền đơn tùy ý (như trong SWRL chẳng hạn) hoặc áp dụng một kết hợp các ký tự (có thể là chồng chéo như trong chương trình logic mô tả). Một ký tự như vậy thường định nghĩa hai miền khác nhau của tập vũ trụ và tương tác của chúng. Trong thiết lập sau này, người ta lấy vũ trụ Herbrand như là miền cho *P*.

Ta cũng thấy trong các tính năng nhóm thứ hai, chương trình logic mô tả và $\mathcal{DL}$*+log* có tên duy nhất (UNA) trong vũ trụ Herbrand. Trong thực tế, $\mathcal{DL}$*+log* đáp ứng UNA trong toàn bộ tri thức, được suy diễn ra bởi cách người ta áp dụng giả định tên chuẩn. Lưu ý, tuy nhiên, $\mathcal{DL}$ *+log* không cam kết Herbrand giải thích của các hằng số trong phần quy định.

Liên quan đến sự tương tác từ ontology (lý thuyết bậc nhất) đến các quy tắc, chúng ta phân biệt được cho dù tính đúng của literal với vị ngữ truyền thống trong một quy tắc phụ thuộc vào việc mô hình xây dựng trên trên một mô hình duy nhất của phần bậc nhất trong một cơ sở tri thức lai, hoặc trên kế thừa từ nhiều mô hình. $\mathcal{DL}$*+log* làm việc trên mô hình duy nhất, trong khi đó chương trình logic mô tả suy luận ra từ nhiều mô hình; trong chương trình logoc mô tả, thông tin từ các lý thuyết bậc nhất được lấy từ các quy tắc chỉ khi một truy vấn được chứng minh từ các thiết lập của các mô hình của phần bậc nhất.

Đối với hướng ngược lại (từ các quy tắc vào phần bậc nhất), tương tác mô hình đơn này được hiểu theo nghĩa là mỗi mô hình $\mathcal{I}$ của phần các quy tắc $\mathcal{P}$ ràng buộc mô hình của phần bậc nhất $\mathcal{T}$ như vậy chỉ các mô hình này sẽ được xem xét trong đó tất cả các vị từ truyền thống có một mức độ lớn hơn trong $\mathcal{I}$. Kế thừa dựa tương tác đơn giản chỉ cần thêm kết luận tích cực về các vị từ truyền thống mà chúng có thể được rút ra từ mô hình của chương trình logic đến phần bậc nhất. Lưu ý rằng điều này có thể làm cho một sự khác biệt, nếu chúng ta có thể có các yếu tố trong các diễn dịch mà không thể được truy cập thông qua hạng thức nền. Ở đây, chỉ có chương trình logic mô tả được hình thành theo nguyên tắc thứ hai, thông qua các thiết bị đặc biệt là nguyên tử logic mô tả dl-atom, chúng thêm các kết luận về các vị từ cổ điển vào các ontology.

Là một tham số cuối cùng nhưng rất quan trọng, chúng ta xem xét đến tính quyết định. Chương trình logic mô tả là quyết định được với điều kiện kiểm tra tính thỏa đối với các cơ sở logic mô tả cơ bản là quyết định được và phần quy tắc là *DL-safe*. $\mathcal{DL}$*+log* cũng đung trong các trường hợp có DL-safety yếu.

4. KẾT LUẬN

Các framework suy diễn tiên tiến cho các ứng dụng web có ngữ nghĩa tương lai cần phải giải quyết các vấn đề với cả các quy tắc và các ontology một cách thống nhất trong các phương pháp tích hợp mà hiện đang không được hỗ trợ tốt. Trong bài viết này, chúng ta đã xem xét một số hệ hình thức dựa trên các quy tắc để làm việc với các cơ sở ontology. Chúng làm việc ở các cấp độ khác nhau trong quá trình tích hợp, từ trình độ thấp đến một mức độ cao, lúc đó một ngữ nghĩa xác thực được đưa ra cho việc tích hợp của các quy tắc và các ontology.

Ngoài các kiến thức cơ bản liên quan đến ontology và các quy tắc, tôi đã trình bày các vấn đề kỹ thuật cơ bản trong việc tích hợp các quy tắc và các ontology và các các phương pháp tiếp cận khác nhau cụ thể là các phương pháp tích hợp chặt chẽ ngữ nghĩa và các phương pháp tích hợp linh hoạt ngữ nghĩa đồng thời đưa ra các ví dụ tiêu biểu là $\mathcal{DL}$*+log* và chương trình logic mô tả cùng các đánh giá, so sánh giữa chúng và các hướng phát triển sau này.

TÀI LIỆU THAM KHẢO

[1]. T. Berners-Lee (1999), *Weaving the Web*, Harper, San Francisco, CA.

[2]. T. Berners-Lee, J. Hendler, O. Lassila (2001), *The Semantic Web*, Scientific American 34–43.

[3]. R. Neches, R. Fikes, T. W. Finin, T. R. Gruber, R. S. Patil, T. E. Senator, and W. R. Swartout. *Enabling technology for knowledge sharing*. AI Magazine, 12(3):36–56, 1991. 1.2

[4]. I. Horrocks, P.F. Patel-Schneider (2003), Reducing OWL entailment to description logic satisfiability, in: *Proceedings ISWC-2003, in: LNCS, vol. 2870, Springer*, pp. 17–29.

[5]. Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, Hans Tompits (2006), Reasoning with Rules and Ontologies, *Lecture Notes in Computer Science*, *vol. 4126*, *Springe*, pp 93-127.

[6]. T. Eiter, G. Ianni, T. Krennwallner, and R. Schindlauer. Exploiting conjunctive queries in description logic programs. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, pages 259–266, 2007.

[7]. T. Eiter, G. Ianni, T. Krennwallner, and R. Schindlauer. Exploiting conjunctive queries in description logic programs. Technical Report INFSYS RR-1843-08-02, Institut fur Informationssysteme, Technische Universitat Wien, A-1040 Vienna, Austria, Mar. 2008. Extended version of the DL'07/ISAIM'08 abstract.

[8]. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. Technical Report INFSYS RR- 1843-07-04, Institut fur Informationssysteme, TU Wien, Mar. 2007. To appear in *Artificial Intelligence*.

[9]. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set program- ming with description logics for the Semantic Web. In *Proceedings KR-2004*, pages 141–151, 2004.

APPROACHES TO INTEGRATING ONTOLOGIES AND RULES IN SEMANTIC WEB

Hoang Nguyen Tuan Minh

Office for Student Affairs, Hue University College of Sciences

Email: hntminh83@yahoo.com

ABSTRACT

For realizing the Semantic Web vision, the integration of different layers of its architecture is a fundamental issue. Rules and ontologies play an important role in ontological layer of the Semantic Web, of which they are used to ascribe meaning and to inference data on the Web. Recently, researchers have been interested in the integration of ontology layer and the rule layer for developing it. Although there have been some proposals to solve this problem, these solutions still meet the different obstacles and have not achieved the desired effect. This article gives a fairly comprehensive overview, provides evaluation and comparision of such solutions in relation to the combination of rules and ontologies in the Semantic Web architecture.

Keyworks: *Description logics, Logic programing, , Ontology, Semantic Web.*