UNIVERSITY OF INFORMATION AND TECHNOLOGY

Báo cáo khóa luận giai đoạn 8-10/2014

hoàn thành bởi

với sự hướng dẫn của

TS. Nguyễn Anh Tuấn

Đề tài khóa luận
tốt nghiệp - sinh viên khóa MMTT2010

thuộc
Chuyên ngành phát triển ứng dụng Web và di động
Khoa mạng máy tính và truyền thông

Ngày 23 tháng 10 năm 2014

Mục lục

For/Dedicated to/To my...

Chương 1

Các nguyên nhân dẫn đến tính thiếu nhất quán trong ontology

1.1 Các khái niệm cần lưu ý[3]

1.1.1 Lớp không thỏa về nghĩa

Khái niệm Unsatisfiable Class

Giải thích Khái niệm nêu trên dùng để chỉ một lớp hay một đối tượng trong một Ontology được định nghĩa không thoả về nghĩa với các phát biểu logic trong ontology hay có thể nói là các phát biểu logic của lớp này mâu thuẫn với nhau.

Ví dụ `Cow`

```
        SubClassOf: Vegetarian
Vegetarian
     SubClassOf: Animal and eats only Plant
DisjointClasses:
     Plant, Animal
```

Giải thích ví dụ Trong ví dụ trên thì MadCow chính là một lớp không hợp lý do trong các phát biểu logic của nó mâu thuẫn với nhau Cow là lớp con của Vegeterian mà Vegeterian chỉ ăn Plant (nghĩa là ngoài Plant, Vegeterian không ăn thứ gi khác) trong khi đó khai báo của lớp MadCow là lớp con của Cow và ăn một số Sheep (Sheep là một lớp con của Animal).

Từ đó việc lý luận có thể đưa ra giả định sai là Sheep cũng có khả năng là một phần của Plant . Điểm quan trọng là Plant và Animal là 2 DisjointClasses, nói cách khác không tồn tại một cá thể nào vừa thuộc lớp Plant và vừa thuộc lớp

Animal. Như vậy trong tất cả các phát biểu logic ở ví dụ trên đã có 2 phát biểu gây mâu thuẫn chính là eats only Plant và eats some Sheep, và chúng làm cho lớp MadCow trở nên bất hợp lý (unsatisfiable).

1.1.2 Ontology không mạch lạc rõ ràng

Khái niệm Incoherent ontology

Giải thích Ontology không mạch lạc rõ ràng dùng để chỉ một ontology hoặc một model có chứa các lớp hoặc các đối tượng không hợp lý (unsatisfiable classes) và điểm quan trọng là các lớp có tính mâu thuẫn đó không được có bất kì cá thể (individual) nào.

Giả sử ta có ontology A chứa các các phát biểu trong ví dụ trên ngoại trừ phát biểu cuối cùng `Individual: Dora type:` MadCow thì ta có thể nói ontology A không mạch lạc rõ ràng do nó chứa lớp có tính mâu thuẫn là MadCow. Chúng ta vẫn có thể sử dụng ontology A miễn là không có phần tử nào thuộc lớp MadCow.

1.1.3 Ontology không nhất quán

Khái niệm Inconsistent Ontology

Giải thích Một ontology không có tính nhất quán dùng để chỉ một ontology chứa ít nhất một lớp hoặc các đối tượng có tính bất hợp lý (unsatisfiable classes) và các lớp hoặc đối tượng này lại có những cá thể thuộc nó.

Như đã thể hiện trong ví dụ đầu tiên thì cá thể Dora thuộc lớp MadCow (Một lớp có tính mâu thuẫn thì không được phép có bất kì cá thể nào), như vậy bất kì ontology nào có những phát biểu trên đều được coi là không nhất quán (inconsistency), đều này đồng nghĩa là ontology đó không thể sử dụng được.

1.2 Các nguyên nhân phổ biến dẫn đến tính thiếu nhất quán (inconsistency)[4]

Các nguyên nhân dẫn đến tính thiếu nhất quán trong ontology gây bởi các lỗi được phân loại thành lỗi gây ra bởi phát biểu ở mức độ lớp (class level - TBox), các lỗi gây ra bởi phát biểu ở mức độ cá thể (instance level - ABox) và lỗi gây ra bởi phát biểu phát biểu liên quan kết hợp ở mức độ lớp và cá thể.

1.2.1 Khởi tạo cá thể cho một lớp không hợp lý (unsatisfiable class) - (TBox + ABox)

- Đưa một cá thể vào một lớp có tính phi lý được xem là nguyên nhân phổ biến nhất gây ra tính thiếu nhất quán trong ontology
- Ví dụ:

```
Individual: Dora type: MadCow
```

- Chúng ta không cần biết đâu là nguyên nhân làm cho `MadCow` trở nên mâu thuẫn, chỉ cần biết là một lớp chứa sự mâu thuẫn thì không nên có bất kì cá thể nào trong đó. Rõ ràng là không có bất kì ontology nào mà cá thể Dora có thể đáp ứng các điều kiện như trong ví dụ đầu tiên, nói cách khác là ontology không tồn tại. Chúng ta phát biểu đó là một ontology không nhất quán.

1.2.2 Khởi tạo cá thể thuộc 2 class được disjoint với nhau (TBox + ABox)

- Đây là một trường hợp dễ bắt gặp vì nó sai ngay trong phát biểu về mặt ngữ nghĩa
- Ví dụ

```
Individual: Dora
Types: Vegetarian, Carnivore
DisjointClasses: Vegetarian, Carnivore
```

- Phát biểu ở cấp độ lớp DisjointClasses(Vegetarian Carnivore) có nghĩa là các lớp được gọi là DisjointClasses là các lớp mà không có chung bất kì một phần tử hay cá thể nào, đồng nghĩa với không có tồn tại bất kì cá thể nào thuộc các lớp đã được disjointed với nhau,do đó phát biểu cấp độ cá thể (ABox) `Dora Types: Vegeterian, Carnivore` là sai.

1.2.3 Các phát biểu phát biểu cho cá thể xung đột với nhau (All Abox)

- Ở trường hợp này thì tương tự như nguyên nhân ở trên nhưng khác ở chỗ là lần này sự mâu thuẫn nằm hết trong các biểu ở cấp độ cá thể (ABox).
- Ví dụ:

```
Individual: Dora
   Types: Vegetarian, not Vegetarian
```

- Dễ dàng nhận thấy được sự mâu thuẫn trong trong phát biểu trên vừa yêu cầu Dora là Vegetarian vừa yêu cầu nó không phải Vegetarian.

1.2.4 Các phát biểu xung đột với dạng phát biểu bao gồm hay một trong (All TBox)

- Phát biểu bao gồm hoặc một trong (oneOf trong ngôn ngữ OWL) cho phép sử dụng các cá thể trong phát biểu cấp độ lớp (TBox), sự kết hợp này có thể dẫn đến sự thiếu nhất quán.

- Lấy ví dụ sau:

```
Class: MyFavouriteCow
   EquivalentTo: {Dora}
Class: AllMyCows
   EquivalentTo: {Dora, Daisy, Patty}
DisjointClasses: MyFavouriteCow, AllMyCows
```

- Phần đầu tiên của các phát biểu trên tất cả các thể thuộc lớp MyFavouriteCow phải tương đương với cá thể tên Dora, nói cách khác là SameIndividual với Dora. Phần thứ hai cũng tương tự tất cả các cá thể thuộc lớp AllMyCows buộc phải tương đương với một trong 3 cá thể tên Dora, Daisy hoặc Patty. Do 2 phát biểu trên chúng ta đã nói Dora thuộc cả 2 lớp MyFavoriteCow và AllMyCows nên mâu thuẫn với phát biểu cuối cùng khi nói 2 lớp này không có chung một cá thể nào. Vì vậy dẫn tới ontology bị thiếu nhất quán (inconsistent).

1.2.5 Không có khả năng khởi tạo bất kì cá thể nào (all TBox)

- Ví dụ:

```
Vegetarian or not Vegetarian
SubClassOf: Cow and not Cow
```

- Dây chỉ là một ví dụ đơn giản để minh hoạ. Thực tế sẽ không ai tạo ra một phát biểu ngớ ngẩn như vậy nhưng nó sẽ có khả năng xảy ra khi phát biểu trên là kết quả suy luận(reasoning) của những phát biểu lớn và phức tạp hơn.

- Có thể giải thích ví dụ trên như sau. Đầu tiên để đáp ứng ý nghĩa dòng đầu tiên yêu cầu cá thể vừa là Vegetarian hoặc không phải Vegetarian - bất kỳ phát biểu nào yêu cầu cá thể thuộc một lớp hoặc không thuộc lớp đó thì phát biểu đó ám chỉ đến tất cả cá thể xuất hiện trong ontology.Dòng thứ hai yêu cầu cá thể vừa

là Cow vừa không phải là Cow, phát biểu này rơi vào một trong các nguyên nhân đã nếu ở trên. Cuối cùng yêu cầu tất cả cá thể vừa là Cow vừa không phải Cow là điều bất hợp lý do đó làm ontology thiếu nhất quán.

Kết luận Trên đây chúng ta đã liệt kê những nguyên nhân phổ biến dẫn đến thiếu nhất quán qua những ví dụ đã được đơn giản hoá tối đa để chung ta dễ nhận ra được dấu hiệu gây lỗi. Trên thực tế nhưng nguyên nhân như khai báo cá thể cho một lớpbất hợp lý (unsatisfiable class) có thể dễ phát hiện bởi người xây dựng ontology, những nguyên nhân khác như liên quan tới định danh (oneOf) thì sẽ khó phát hiện được hơn.

Chương 2

Giải pháp để sửa chữa inconsistent ontology

- Như đã được đề cập trong chương 1, trong các nguyên nhân dẫn đến tính thiếu nhất quán(inconsistency) trong ontology thì **unsatisfiable class** (lớp không thỏa về nghĩa) là nguyên nhân nếu có thể được phát hiện sớm để sửa lại hoặc loại bỏ các phát biểu gây mâu thuẫn thì giúp cho ontology đạt được sự nhất quán.

- Đã có rất nhiều nghiên cứu thành công trong việc tìm và phát hiện lỗi (các phát biểu mâu thuẫn) trong ontology. Trong đó có một nghiên cứu nổi bật[1], không chỉ có khả năng phát hiện gần như chính xác các nguyên nhân gây lỗi mà còn được đưa ra các giải pháp tối ưu* để sửa lỗi. Nghiên cứu này đã được ứng dụng để đưa ra các giải thích về các lớp không thỏa về nghĩa (unsatisfiable classes) trong bộ thự viện lập trình ontology thông dụng hiện nay là OWL-API[2]. Sau đây chúng em xin được trình bày lại những điểm quan trọng trong nghiên cứu vừa được đề cập**.

** Tối ưu có nghĩa là hạn chế tối đa các thay đổi về ý nghĩa mà việc xóa hoặc thay đổi phát biểu mâu thuẫn có thể gây ra cho các phát biểu khác (other axioms) trong ontology.*

*** Mọi quan điểm và ý tưởng trình bày ở phần sau của Chương 2 đều thuộc sở hữu của các tác giả bài báo[1]. Chúng em chỉ trình bày lại sau khi đã đọc và hiểu được ý tưởng chính của bài báo.*

2.1 Mục tiêu của việc debuging ontology

Mục tiêu chính của việc debuging ontology gồm hai điểm chính yếu. Thứ nhất cho một ontology có số lượng lớn các lớp không thỏa về nghĩa(unsatisfiable), tìm và nhận dạng được nguồn gốc của lớp gây ra mâu thuẫn và các lớp bị ảnh hưởng bởi sự mâu thuẫn đó. Thứ hai cho biết trước tên một lớp cụ thể không thỏa về nghĩa, tách và trình bày cho người sử dụng ontology một tập hợp tối tiểu các phát biểu (minimal set of axioms) từ ontology chịu trách nghiệm gây ra sự mâu thuẫn về nghĩa.
Nghiên cứu của họ đã đóng góp những kỹ thuật nổi bật đã được ứng dụng vào trong các thư việc OWL-API[2] và Pellet Reasoner[3] như:

- Tăng cường những thông tin hữu ích cho quá trình sử lỗi các lớp không thỏa về nghĩa bằng cách chỉnh sửa giải thuật của họ để thu được một phần của những phát biểu gây lỗi.
- Đưa ra một kỹ thuật để đưa ra các giải pháp sửa lỗi tự động dựa trên các kỹ thuật dùng để xếp hạng các phát biểu gây lỗi và một giải thuật Reiter's Hitting Set đã được chỉnh sửa. Bên cạnh đó, họ cũng đưa ra các kỹ thuật để viết lại các phát biểu gây lỗi.
- Họ cũng đã áp dụng thành công kết quả nghiên cứu của họ vào trong Swoop Ontology Editor[*] và OWL-API

2.2 Các bước sửa chữa các phát biểu bị lỗi

2.2.1 Minimal Unsatisfiability Preserving Sub-TBoxes (MUPS)

Khái niệm MUPS lần đầu được giới thiệu trong[6]. Một MUPS của một lớp nguyên tử (*atomic class or concept*) A là một phần tối tiểu của Knowledge Base mà trong đó A không thỏa nghĩa (*unsatisfiable*). Mỗi lớp không thỏa nghĩa sẽ có nhiều MUPS khác nhau trong một ontology. Tìm tất cả MUPS của một lớp không thỏa nghĩa là một nhiệm vụ cực kì khó đứng ở khía cạnh debuging, từ đó để sửa lại một lớp không thỏa nghĩa(*unsatisfiable class*) chúng ta cần loại bỏ tối thiểu ít nhất một phát biểu từ từng tập các phát biểu trong MUPS diện cho unsatisfiable class đó.

* *Rất tiếc là Swoop Editor đã không còn cập nhật phiên bản nào mới kể từ năm 2007, nhưng kết quả nghiên cứu của họ cũng đã được ứng dụng vào trong việc tạo ra các giải thích cho unsatisfiable class trong thư viện OWL-API[2].*

2.2.2 Tìm MUPS của một lớp không thỏa nghĩa

Axiom Pinpointing service Một thành phần quan trọng trong giải pháp debugging dùng để tìm và trích xuất MUPS từ một lớp không thỏa nghĩa. Họ cũng đã nâng cấp chức năng này nhằm xác định được chính xác những phần phát biểu gây lỗi, họ định nghĩa MUPS tìm được là *precise* MUPS.
Ý tưởng của kĩ thuật này có thể được mô tả khái quát như sau(chi tiết hơn về cách áp dụng xem[8]). Mục tiêu là nhận diện những phần có liên quan của các phát biểu, nên họ đã định nghĩa một hàm nhằm chia nhỏ các phát biểu trong KB **K** thành những phát biểu nhỏ hơn để thu được KB K_s có nghĩa tương đương với KB **K**, với số lượng các phát biểu được chia nhỏ nhiều nhất có thể.
Ví dụ: $A \subseteq C \cap$D tương đương với $A \subseteq$C, $A \subseteq$D
Trong một vài trường hợp, kĩ thuật này bắt buộc phải giới thiệu ra một tên lớp mới, chỉ với mục đích là chia các phát biểu thành những mảnh nhỏ hơn.
Ví dụ: $A \subseteq \exists$R.($C \cap$D) không tương đương với $A \subseteq \exists$R.C, $A \subseteq \exists$R.D
Để chia nhỏ phát biểu trên chúng ta sẽ giới thiệu một tên lớp mới, gọi là *E*.
Như vậy ta có: $A \subseteq \exists$R.($C \cap$D) sẽ tương đương với $A \subseteq \exists$R.E, $E \subseteq$C, $E \subseteq$D, $C \cap$D $\subseteq$E Cuối cùng nhiệm vụ tìm kiếm *precise* MUPS của lớp không thỏa nghĩa trong KB **K** giảm xuống thành vấn đề tìm MUPS trong những phiên bản đã được tách nhỏ trong KB K_s. Chi tiết hơn về kĩ thuật này xin đọc[8].

2.2.3 Chiến thuật xếp hạng các phát biểu (*Axioms*)

Đây là một phần chính yếu trong tiến trình chỉnh sửa lại các phát biểu gây lỗi, quyết định xem nên loại bỏ phát biểu nào từ MUPS để sửa được lớp không thỏa nghĩa. Với mục tiêu này

2.2.4 Đưa ra các nguyên nhân gây lỗi chính xác bằng giải thuật HST của Reiter

Qua các bước trên, họ đã đưa ra một quy trình để tìm được ...

Tìm MUPS bằng giải thuật HST của Reiter Một kỹ thuật tối ưu hơn kỹ thuật **Axiom Pinpointing** đã được đề cập ở mục trên nhằm tìm kiếm MUPS của một unsatisfiable class, họ đã sử dụng giải thuật Hitting Set của Reiter[7], nhằm để xác định căn nguyên(*root cause*) của một vấn đề từ một bộ(*collection*) gồm nhiều tập hợp đụng độ chứa các nguyên nhân dẫn tới vấn đề, giải thuật này sẽ tạo ra những tập tối tiểu (*minimal hitting set*) chứa các nguyên nhân gây ra vấn đề. Một tập hợp đụng độ (*hitting*

set) trong một bộ **C** các tập hợp là tập hợp giao (có chung phần tử) với từng tập hợp trong **C**. Một tập hợp đụng độ là tối tiểu nếu không có bất kì tập con nào của nó lại là một tập đụng độ cho **C**. Để áp dụng vào trường hợp này khi chúng ta đã có một bộ chứa tập hợp chứa các nguyên nhân gây lỗi(*MUPS*), họ áp dụng giải thuật này để tạo ra các tập đụng độ từ MUPS - ý tưởng ở đây là loại bỏ tất cả các phát biểu trong tập đụng độ tối tiểu sẽ giúp loại bỏ từng phát biểu gây lỗi trên từng tập phát biểu của MUPS và cuối cùng giúp cho sửa chữa được unsatisfiable class.

Chương 3

Tài liệu tham khảo

1. Aditya Kalyanpur, Bijan Parsia, Evren Sirin , Bernardo Cuenca-Grau.Repairing Unsatisfiable Concepts in OWL Ontologies, 2007. Available online at http://www.cs.ox.ac.uk/people/bernardo.cuencagrau/publications/repair.pdf

2. The OWL API. The OWL API is a Java API for creating, parsing, manipulating and serialising OWL Ontologies. Available online at https://github.com/owlcs/owlapi

3. Pellet - OWL 2 Reasoner for Java.Pellet is an OWL 2 reasoner. Pellet provides standard and cutting-edge reasoning services for OWL ontologies. Available online at http://clarkparsia.com/pellet/

4. Robert Stevens. Ontogenesis, (I can't get no) satisfiability Available online at http://ontogenesis.knowledgeblog.org/1329

5. Samantha Bail.Ontogenesis, Common reasons for ontology inconsistency.Available online at http://ontogenesis.knowledgeblog.org/1343

6. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Proceedings of IJCAI, 2003.

7. R. Reiter. A theory of diagnosis from first principles. 1987. Artificial Intelligence 32:57-95.

8. A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and E. Sirin. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in SHOIN (owl-dl). Technical report, UMIACS, 2005-66, 2006. Technical Report

Phụ lục A

An Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus at pulvinar nisi. Phasellus hendrerit, diam placerat interdum iaculis, mauris justo cursus risus, in viverra purus eros at ligula. Ut metus justo, consequat a tristique posuere, laoreet nec nibh. Etiam et scelerisque mauris. Phasellus vel massa magna. Ut non neque id tortor pharetra bibendum vitae sit amet nisi. Duis nec quam quam, sed euismod justo. Pellentesque eu tellus vitae ante tempus malesuada. Nunc accumsan, quam in congue consequat, lectus lectus dapibus erat, id aliquet urna neque at massa. Nulla facilisi. Morbi ullamcorper eleifend posuere. Donec libero leo, faucibus nec bibendum at, mattis et urna. Proin consectetur, nunc ut imperdiet lobortis, magna neque tincidunt lectus, id iaculis nisi justo id nibh. Pellentesque vel sem in erat vulputate faucibus molestie ut lorem.

Quisque tristique urna in lorem laoreet at laoreet quam congue. Donec dolor turpis, blandit non imperdiet aliquet, blandit et felis. In lorem nisi, pretium sit amet vestibulum sed, tempus et sem. Proin non ante turpis. Nulla imperdiet fringilla convallis. Vivamus vel bibendum nisl. Pellentesque justo lectus, molestie vel luctus sed, lobortis in libero. Nulla facilisi. Aliquam erat volutpat. Suspendisse vitae nunc nunc. Sed aliquet est suscipit sapien rhoncus non adipiscing nibh consequat. Aliquam metus urna, faucibus eu vulputate non, luctus eu justo.

Donec urna leo, vulputate vitae porta eu, vehicula blandit libero. Phasellus eget massa et leo condimentum mollis. Nullam molestie, justo at pellentesque vulputate, sapien velit ornare diam, nec gravida lacus augue non diam. Integer mattis lacus id libero ultrices sit amet mollis neque molestie. Integer ut leo eget mi volutpat congue. Vivamus sodales, turpis id venenatis placerat, tellus purus adipiscing magna, eu aliquam nibh dolor id nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed cursus convallis quam nec vehicula. Sed vulputate neque eget odio fringilla ac sodales urna feugiat.

Phasellus nisi quam, volutpat non ullamcorper eget, congue fringilla leo. Cras et erat et nibh placerat commodo id ornare est. Nulla facilisi. Aenean pulvinar scelerisque eros eget interdum. Nunc pulvinar magna ut felis varius in hendrerit dolor accumsan. Nunc pellentesque magna quis magna bibendum non laoreet erat tincidunt. Nulla facilisi.

Duis eget massa sem, gravida interdum ipsum. Nulla nunc nisl, hendrerit sit amet commodo vel, varius id tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor est. Suspendisse ultrices tincidunt metus eget accumsan. Nullam facilisis, justo vitae convallis sollicitudin, eros augue malesuada metus, nec sagittis diam nibh ut sapien. Duis blandit lectus vitae lorem aliquam nec euismod nisi volutpat. Vestibulum ornare dictum tortor, at faucibus justo tempor non. Nulla facilisi. Cras non massa nunc, eget euismod purus. Nunc metus ipsum, euismod a consectetur vel, hendrerit nec nunc.