**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**\_\_\_ \*\*\* \_\_\_**



**MÔN HỌC  
Các hệ Cơ sở Tri thức**

**Báo cáo đồ án**

Đề tài: **Xây dựng hệ chuyên gia tư vấn mua máy tính**

**GVLT**: Nguyễn Thị Ngọc Diễm

**Thành viên**:

* **Nguyễn Trần Minh Tân** 13520747
* **Lê Thị Tuyết Mai** 13520489
* **Nguyễn Minh Phúc** 13520638

**Hồ Chí Minh, Ngày 20 Tháng 1 Năm 2016**

**LỜI CẢM ƠN**

Nhằm xây dựng một hệ thống tư vấn mua máy tính cho những khách hàng phục vụ nhu cầu công việc, hệ thông tư vấn mua máy tính ra đời.

Trước tiên nhóm em xin gửi lời cám ơn chân thành sâu sắc tới cô Nguyễn Thị Ngọc Diễm cô đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn nhóm em trong suốt quá trình làm đồ án. Trong thời gian làm việc với cô, nhóm em không ngừng tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu khoa học nghiêm túc, hiệu quả, đây là những điều rất cần thiết cho nhóm em trong quá trình học tập và làm việc sau này. Sau cùng xin gửi lời cảm ơn chân thành tới các thành viên trong nhómđã tích cực đóng góp ý kiến và giúp đỡ nhau trong quá trình học tâp, nghiên cứu và hoàn thành đồ án.

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1. ĐẶT VẤN ĐỀ

Trước hết, thực tế cho thấy một điều rằng thế giới ngày nay nói chung và lĩnh Khoa

học Máy tính nói riêng đang thay đổi một cách rất mạnh mẽ. Những vấn đề tưởng chừng như không thể giải quyết bằng một người, hai người, thì bây giờ chỉ cần một chiếc máy tính là có thể giải quyết một cách nhanh chóng. Không dừng lại ở đó, những lĩnh vực mà trước đây Khoa học Máy tính không thể chen chân vào thì nay cũng đã bắt đầu thấy được sự hiện diện theo một cách rất tích cực. Tại sao lại như thế, tất cả nằm ở con người. Nhân loại rất vất vả trong việc tìm ra cách để lưu trữ thông tin, nhưng khi đã lưu trữ được rồi, chúng ta lại có thêm tham vọng đó là giúp chiếc máy tính có thể dựa vào những nguồn thông tin lưu trữ, sự kiện có sẵn đó mà có thể có khả năng lập luận, suy luận để rút ra được những kết luận từ những nguồn dữ liệu như trên.

Trí tuệ nhân tạo (Artificial Intelligence) ra đời như là một điều tất yếu đi cùng với sự phát triển của Khoa học Máy tính, và ứng dụng của chúng thì được sử dụng rộng rãi. Hệ Chuyên Gia (Expert System) là một hệ thống dựa trên tri thức, là một chương trình máy tính chứa một số tri thức đặc thù của một hoặc nhiều chuyên gia là con người về một chủ

đề cụ thể, giúp giải quyết các vấn đề phức tạp trong cuộc sống. Tại đây nhóm quyết định vẫn đề tìm hiểu là tư vấn người dùng mua máy tính.

Trên thị trường hiện nay muốn tìm ra một sảng phẩm tốt, bền và có thường hiệu, hay những sản phẩm mới được ra có thành phần và chất lượng cao hơn những sản phẩm khác. Đối với hàng ngàn hàng triệu sinh viên có lẽ máy tính là cái mà mọi người ai cũng sở hữu cho mình, giá trị của chiếc máy tính rất lớn, có thể với nhiều sinh viên nó đơn giản không có giá trị lớn, nhưng đối với một lượng lớn sinh viên khác nó là tài sản quý báu nhất của sinh viên. Vì vậy để mua một chiếc máy tính ngon hay nói cách khác là ‘xịn’ có cấu hình ngon , có chức năng tốt đã những vấn đề sinh viên tìm hiểu. Sinh viên tìm hiểu ở đâu ? Sinh viên muốn mua máy tính có thể sẽ lên seach như sau : những chiếc máy tính mới nhất hiện nay ? Gía thành của máy tính có cao không ? Ram bao nhiêu thì dùng được ? Chơi game được ? Màn hình ? ….. Nhiều thông tin mà sinh viên cầN.

Vậy những thông tin được lấy đó họ có thể tổng hợp và tư vấn cho bạn 1 chiếc má tính phù hợp với yêu cầu của bạn không?

Vì vậy nhóm chúng em đề xuất xây dựng chương trình tư vấn mua máy tính .

1. MỤC TIÊU NGHIÊN CỨU

Dựa trên những phân tích ở trên, trước hết mục tiêu là muốn tạo ra sự tiện lợi trong

Trong việc cọn mua máy tính cho hợp ý mình vừa tiến. xây dựng một chương trình tìm kiếm tư vấn để khách hàng mua máy tính theo ý mình một cách chuyên nghiệp nhất có thể.

Tìm hiểu các phương pháp biểu diễn tri thức, các kỹ thuật suy diễn dùng trong hệ chuyên gia cho việc tư vấn mua máy tính một cách hiệu quả nhất . Cuối cùng là xây dựng được một hệ thống như mong đợi.

1. ĐỐI TƯỢNG NGHIÊN CỨU

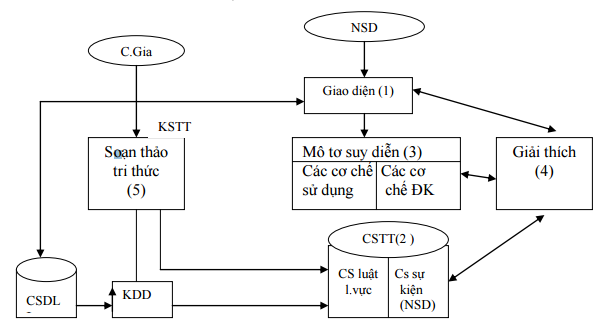
Đề tài sẽ xoay quanh các vấn đề về miền tri thức máy tính, tìm ra những chiếc máy tính mà tư vấn theo yêu cầu phù hợp nhất có thể và dùng các kỹ thuật trong việc tổ chức tri thức, lưu trữ tri thức và suy diễn.

1. GIỚI HẠN VÀ PHẠM VI NGHIÊN CỨU

Vì thời gian thực hiện đề tài có hạn nên nhóm chúng em xin phép chọn miền tri thức là laptop và PC.

Giới hạn nghiên cứu của bọn em ở mức tư vấn người dùng mua laptop cho phù hợp với yêu cầu của người dùng.

1. PHƯƠNG PHÁP NGHIÊN CỨU
   * Tìm hiểu tài liệu: nghiên cứu tài liệu, thông tin liên quan, tri thức từ chuyên gia, tìm nguồn tài liệu liên quan đến vấn đề đang thực thi.
   * Phân tích dữ liệu : Tri thức lấy về ở dạng thô, phân loại sơ bộ tri thức
   * Lưu trữ tri thức: Phân loại tri thức xong chúng ta thực hiện việc lưu trữ tri thức theo mô hình cơ sở tri thức mà chúng ta đã thiết kế
   * Suy luận tìm ra kết quả dựa vào input của người dùng Dựa vào tri thức thức được lưu trữ dùng chiến lược suy diễn để tìm ra kết quả.
2. NỘI DUNG THỰC HIỆN



CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

1. HỆ CƠ SỞ TRI THỨC

2.1.1 Cơ sở tri thức.

Tri thức là những gì mà một người có thể biết và hiểu được. Tri thức có thể được phân loại thành tri thức có cấu trúc hoặc không có cấu trúc, tri thức rõ ràng hay là tri thức ngụ ý, không rõ ràng. Những gì mà chúng ta biết được thường là các tri thức rõ ràng. Tri thức không có cấu trúc mà vẫn hiểu được, nhưng không được phát biểu rõ ràng là các tri thức ngầm ý, tri thức không rõ. Khi tri thức được tổ chức để có thể chia sẻ thì lúc đó tri thức được gọi là tri thức có cấu trúc. Để có thể chuyển đổi từ tri thức không rõ ràng sang tri thức rõ ràng thì tri thức đó cần phải được cấu trúc hóa và định dạng lại.

Phân loại tri thức: như đã đề cập ở trên, tri thức có nhiều loại tùy thuộc vào tính chất cấu trúc và tính chất rõ ràng của tri thức. Tri thức có thể được phân loại thành các loại tri thức sau, đây là các loại tri thức thường gặp trong thực tế:

Tri thức có 2 dạng tồn tại chính là tri thức ẩn và tri thức hiện

* Tri thức hiện là những tri thức được giải thích và mã hóa dưới dạng văn bản, tài liệu, âm thanh, phim, ảnh,… thông qua ngôn ngữ có lời hoặc không lời, nguyên tắc hệ thống, chương trình máy tính, chuẩn mực hay các phương tiện khác. Đây là những tri thức đã được thể hiện ra ngoài và dễ dàng chuyển giao, thường được tiếp nhận qua hệ thống giáo dục và đào tạo chính quy.
* Tri thức ẩn là những tri thức thu được từ sự trải nghiệm thực tế, dạng tri thức này thường ẩn trong mỗi cá nhân và rất khó "mã hóa" và chuyển giao, thường bao gồm: niềm tin, giá trị, kinh nghiệm, bí quyết, kỹ năng... VD: Trong bóng đá, các cầu thủ chuyên nghiệp có khả năng cảm nhận bóng rất tốt. Đây là một dạng tri thức ẩn, nó nằm trong mỗi cầu thủ. Nó không thể "mã hóa" thành văn bản, không thể chuyển giao, mà người ta chỉ có thể có bằng cách tự mình luyện tập.

Tùy thuộc vào ứng dụng cụ thể mà tri thức được biểu diễn theo những phương thức khác nhau. Ứng với mỗi dạng biểu diễn đó có một cơ chế để xử lý tri thức đó.

Hệ cơ sở tri thức là một tập hợp các cơ sở lập luận, các luật, các qui trình, thủ tục

được tổ chức thành các lược đồ (giản đồ). Đó là tập hợp của tất cả các thông tin cũng như tất cả kiến thức về một lĩnh vực cụ thể nào đó.

Quá trình thu thập tri thức được gọi là quá trình tiếp nhận tri thức và định dạng tri thức được thực hiện từ nhiều nguồn khác nhau, đặc biệt là từ các chuyên gianh trong các lĩnh vực cụ thể. Công việc này là một trong những bước quan trọng và thường mất rất nhiều thời gian cũng như công sức trong quá trình xây dựng một hệ cơ sở tri thức được sử dụng trong hệ chuyên gia. Trong quá trình phát triển một hệ chuyên gia, các kỹ sư tri thức, những nhà chuyên môn trong lĩnh vực trí tuệ nhân tạo, có nhiệm vụ thu thập kiến thức từ các chuyên gia thuộc lĩnh vực chuyên ngành, sau đó “sao chép” các tri thức đó vào cơ sở tri thức và diễn đạt các tri thức đó dưới dạng có thể dùng được trong hệ chuyên gia.

2.1.2 Hệ chuyên gia

Hệ chuyên gia, còn gọi làhệ thống dựa tri thức, là một[chương trình máy tính](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) chứa một số tri thức đặc thù của một hoặc nhiều chuyên gia con người về một chủ đề cụ thể nào đó. Các chương trình thuộc loại này đã được phát triển từ các[thập niên 1960](https://vi.wikipedia.org/wiki/Th%E1%BA%ADp_ni%C3%AAn_1960) và [1970](https://vi.wikipedia.org/wiki/Th%E1%BA%ADp_ni%C3%AAn_1970), và trở thành ứng dụng [thương mại](https://vi.wikipedia.org/wiki/Th%C6%B0%C6%A1ng_m%E1%BA%A1i) từ [thập niên 1980](https://vi.wikipedia.org/wiki/Th%E1%BA%ADp_ni%C3%AAn_1980). Dạng phổ biến nhất của hệ chuyên gia là một chương trình gồm một tập[**luật**](https://vi.wikipedia.org/w/index.php?title=Lu%E1%BA%ADt_(l%C3%B4gic)&action=edit&redlink=1)phân tích thông tin (thường được cung cấp bởi người sử dụng hệ thống) về một lớp vấn đề cụ thể, cũng như đưa ra các phân tích về các vấn đề đó, và tùy theo thiết kế chương trình mà đưa lời khuyên về trình tự các hành động cần thực hiện để giải quyết vấn đề. Đây là một hệ thống sử dụng các khả năng lập luận để đạt tới các kết luận.

Hệ chuyên gia sử dụng các tri thức của những chuyên gia để giải quyết các vấn đề( bài toán) khác nhau thuộc mọi lĩnh vực.

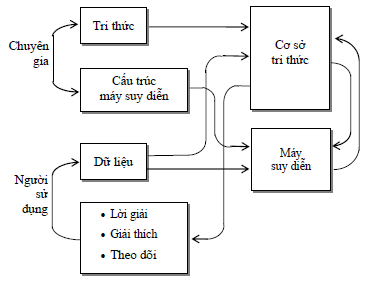
Tri thức ( knowledge ) trong hệ chuyên gia phản ánh sự tinh thông được thu thập từ các chuyên gia hay các nhà bác học, từ sách vở, tạp chí, từ internet với nguồn thông tin vô cùng lớn. Các thuật ngữ hệ chuyên gia, hệ thống dựa trên tri thức ( knowledge-

19

based system ) hay hệ chuyên gia dựa trên tri thức( knowledge-based expert system ) thường có cùng nghĩa.

Một hệ chuyên gia gồm nhiều thành phần nhưng chú trọng 2 phần chính đó là: Cơ sở tri thức(knowledge base) và động cơ suy diễn (inference engine). Cơ sở tri thức chứa các tri thức để từ đó, động cơ suy diễn tạo ra câu trả lời.

Người sử dụng ( user ) cung cấp sự kiện (facts) là những gì đã biết, đã có thật hay những thông tin có ích cho hệ chuyên gia và nhận được những câu trả lời là những lời khuyên hay những gợi ý đúng đắn.



Việc xây dựng một hệ chuyên gia được hiểu như là việc ứng dụng các kiến thức đã học , kiến thức khoa học vào việc vận hành và bảo trì tri thức, người làm được những việc như trên chính là kỹ sư tri thức (Knowledge Engineer). Người kỹ sư tri thức phải đảm bảo rằng máy tính luôn có đủ tri thức để giải quyết một vấn đề nào đó. Người kỹ sư tri thức phải chọn một hoặc nhiều hình thức miêu tả các tri thức được yêu cầu dưới dạng các mô hình kí tự trong bộ nhớ của máy tính và như vậy anh ta phải chọn một phương pháp biểu

20

diễn tri thức. Kỹ sư tri thức phải đảm bảo rằng máy tính có thể sử dụng tri thức một cách có hiệu quả bằng việc chọn ra một trong các phương pháp biểu diễn.

Các đặc tính hệ chuyên gia

1. Tách biệt giữa tri thực và suy diễn

* Cơ sở tri thức và duy diễn được tách rời. Phân tách cơ sở tri thức và mô tơ suy diễn có giá trị trong hệ chuyên gia, đảm bảo tính độc lập trong việc mã hóa tri thức và xử lý tri thức đó.
* Phân tách tri thức ra khỏi động cơ suy diễn để tạo điều kiện biểu diễn tri thực một cách tự nhiên hơn.
* Cơ sở tri thức được tách biệt khỏi cấu trúc điều khiển cấp thấp của chương trình, những người phát triển hệ chuyên gia có thể tập trung một cách trức tiếp vào việc nắm bắt và tổ chức giải quyết vấn đề hơn là việc thiwcj hiện các chi tiết trong việc cài đặt máy tính.
* Sự tách biệt cho phép thay đổi một phần cơ sở tri thực mà không ảnh hưởng lớn đến các phần khác của chương trình.
* Sự tách biệt cho phép thay đổi một phần cơ sở tri thức mà không ảnh hưởng lớn đến các phần khác của chương trình
* Sự tách biệt này cho phép một phần mềm điều khiển và giao tiếp, có thể sử dụng nhiều hệ thống khác nhau/

Ví dụ : Xây dựng một Shell. Các nhà phát triển xây dựng một chương trình cốt lõi của hệ thống. Sau đó tạo ra cơ sở tri thức để giải quyết bài toán.

* Đây là một đặc tính nổi bật trong hệ chuyên gia khác so với phần mềm thông thường.

1. Tri thức chuyên gia

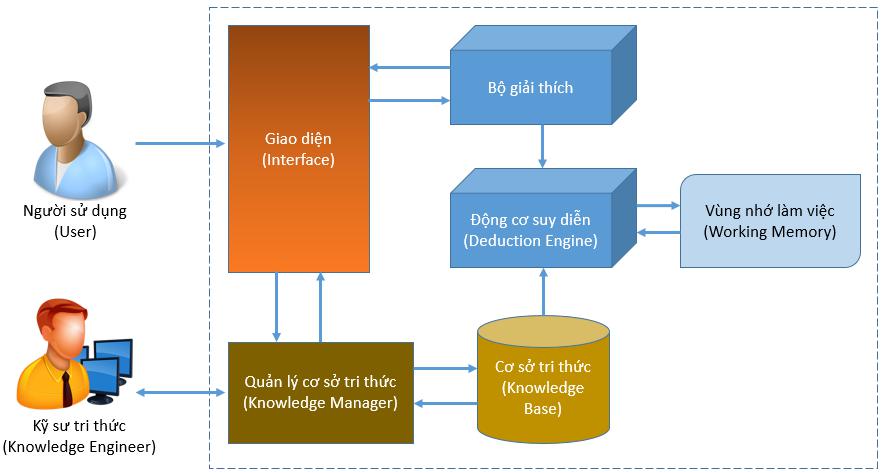
* Đặc điểm nổi bật hệ chuyên gia là khả năng thu thập tri thức của chuyên gia. Tri thức bao gồm tri thức về lĩnh vực và tri thức kĩ năng giải quyết vấn đề. Các tri thức hu được từ chuyên gia không nhất thiết phải là các ý tưởng sang chói hay độc đáo mà đặc biệt và sâu về lĩnh vực cụ thể.

1. Lập luận may rủi

* Các chuyên gia thường dùng kinh nghiệm để giả đúng và hiệu quả baiftoans đang xét. Qua kinh nghiệm đã dùng, họ hiểu vấn đề một cách thực tế và lưu trữ dưới dạng may rủi. chiến lược may rủi được dùng trong hệ chuyên gia để giúp người ta đi nhanh đến giải pháp. Trong hệ chuyên gia chiến lực lập luận may rủi không giống như các thủ tục chính xác của chương trình bình thường.

1. Lập luận không chính xác

Hệ chuyên gia được coi là thành công trong ứng dụng cần đến lập luận không chính xác. Những loại ứng dụng này được đặc trưng bằng thông tin không chắc chắn, nhọc nhằng. Trong thực tế thường xảy ra chẳng hạn như bác sĩ khám bện cho bệnh nhân vào cấp cứu trong hoàn cảnh không có nhiều thông tin về bệnh nhân.



Việc xây dựng một hệ chuyên gia được hiểu như là việc ứng dụng các kiến thức đã học , kiến thức khoa học vào việc vận hành và bảo trì tri thức, người làm được những việc như trên chính là kỹ sư tri thức (Knowledge Engineer). Người kỹ sư tri thức phải đảm bảo rằng máy tính luôn có đủ tri thức để giải quyết một vấn đề nào đó. Người kỹ sư tri thức phải chọn một hoặc nhiều hình thức miêu tả các tri thức được yêu cầu dưới dạng các mô hình kí tự trong bộ nhớ của máy tính và như vậy anh ta phải chọn một phương pháp biểu

20

diễn tri thức. Kỹ sư tri thức phải đảm bảo rằng máy tính có thể sử dụng tri thức một cách có hiệu quả bằng việc chọn ra một trong các phương pháp biểu diễn.

**2.1.3 Các phương pháp biểu diễn tri thức**

1. Thể hiện tri thức bằng cặp ba đối tượng – thuộc tính giá trị
2. Thể hiện các sự kiện không chắn chắn

Trong hệ chuyên gia các sự kiện thường không khẳng định chính xác đúng hay sai và độ chắc chắn tuyệt đối. Đối với các sự kiện chắc chắn người tà không hoàn toàn biết rõ. Không có gì đảm bảo một sự kiện đúng. Vì vậy người ta dùng khái niệm “ mức độ tin cây” vào sự kiện hay viết tắt là cf ( certainty Factor )

Các sự kiện mờ

Trong thực tế cho thấy rằng thể hiện các vấn đề của thế giới thực đôi khi cần dùng đến thuật ngữ nhập nhằng. Chẳng hạn với câu ông ấy “ cao “ là không rõ rang. Không rõ rang vì không xác định được “ cao “ có ý nghĩa gì. Các thuật ngữ nhập nhằng được thể hiện trong tập mờ. Khi đó ta có tập mờ thể hiện chiều caoo của con người.

1. Thể hiện tri thức nhờ các luật

Các sự kiện được cung cấp có ý nghĩa rất lớn đối với hoạt dộng của hệ chuyên gia. Các sự kiện này cho phép hệ thống

2.1.4 Quy trình xây dựng hệ chuyên gia:

Giai đoạn 1:

* Xác định miền tri thức.
* Xác định phạm vi vấn đề.
* Phân tích nhu cầu, yêu cầu.

Giai đoạn 2:

 Thu thập và phân loại tri thức.

Giai đoạn 3: Thiết kế cơ sở tri thức

* Biểu diễn tri thức và mô hình hóa dữ liệu.
* Tổ chức lưu trữ tri thức.

Giai đoạn 4: Thiết kế bộ suy diễn

* Xây dựng mô hình hóa bài toán.
* Phân loại và hợp nhất sự kiện.
* Chiến lược suy diễn.

Giai đoạn 5: Thiết kế giao diện người dùng

Giai đoạn 6: Cài đặt

Giai đoạn 7: Kiểm tra, thử nghiệm

Cơ sở tri thức của hệ chuyên gia bao gồm cả tri thức thực tế và tri thức heuristic. Tri thức thực tế là tri thức chuyên ngành mà được phổ biến và chia sẻ trong phạm vi rộng, có thể tìm thấy dễ dàng trong sách vở, internet. Nói chúng là tri thức này được chấp nhận dựa trên các kiến thức đúng đắn trong từng lĩnh vực cụ thể. Còn tri thức heuristic mang ít tính nghiêm ngặt hơn, dựa nhiều vào kinh nghiệm và sự phán đoán hơn. Trái với tri thức thực tế, tri thức heuristic ít khi được thảo luận và mang đậm tính cá nhân. Nó là kiến thức rút ra từ việc thực hành và phán đoán tốt và sự lập luận chặt hẽ trong từng lĩnh vực.

22

Biểu diễn tri thức là mô hình hóa và tổ chức tri thức. Ở hệ thống này nhóm đã sử dụng hệ luật dẫn để tổ chức lưu trữ tri thức vì nó hỗ trợ rất tốt.

## 2.2 Ontology

### 2.2.1 Khái niệm Ontology

Khoa học máy tính: Ontology là một mô hình dữ liệu biểu diễn một lĩnh vực và được sử dụng để suy luận về các đối tượng trong lĩnh vực đó và mối quan hệ giữa chúng.

Quan điểm Gruber (1993): “Ontology là một “đặc tả” “tường minh” của một sự khái niệm hóa” (An ontology is an explicit specification of a conceptualization).

**Kết luận**: Ontology là phương thức biểu diễn tri thức nhằm biểu diễn tất cả các khái niệm cùng với các mối quan hệ giữa chúng trong 1 lĩnh vực nào đó. Các khái niệm và quan hệ này hình thành nên bộ từ vựng dung chung cho những người có nhu cầu chia sẻ thông tin với nhau trên mạng.

### 2.2.2 Các thành phần của Ontology

Các thành phần phổ biến của ontology bao gồm:

* Các lớp (Classes) - Khái niệm: các khái niệm về các đối tượng hoặc các loại vật chất
* Các cá thể (Individuals hay Instance) - Thể hiện: các đối tượng (các đối tượng cơ bản)
* Các thuộc tính (Properties): các đặc tính, đặc trưng, các đặc điểm hoặc các tham số mà các đối tượng có thể có.
* Các mối quan hệ (Relations): các cách để các lớp, các thực thể có thể được liên kết tới các lớp (hoặc thực thể) khác.
* Các hàm (Functions): Là một loại thuộc tính hay quan hệ đặc biệt, trong đó phần tử thứ n là duy nhất đối với n-1 phần tử còn lại.
* Các giới hạn: các mô tả được phát biểu chính xác và hợp lệ.
* Các luật (Rules): được phát biểu dưới dạng if – then. Với mỗi một tập luật R sẽ có 1 tập đỉnh ứng với luật đó. Kí hiệu Nodes(R).
* Các tiền đề (Axioms): Biểu diễn các phát biểu luôn đúng mà không cần phải chứng minh hay giải thích.
* Các sự kiện (Facts): Các thay đổi của các thuộc tính hoặc các mối quan hệ

**Các thành phần chính**

#### Các lớp (Classes)

Là trung tâm của hầu hết các ontology.

Mô tả các khái niệm trong miền lĩnh vực.

Các lớp thường được tổ chức phân cấp và áp dụng kỹ thuật thừa kế.

Một lớp có thể có các lớp con biểu diễn khái niệm cụ thể hơn so với lớp cha.

#### Các cá thể (Individual)

Biểu diễn các phần tử riêng biệt của khái niệm.

Là các thành phần cơ bản, nền tảng của 1 Ontology.

Là sự thể hiện của một lớp. Mỗi thể hiện của lớp biểu diễn một sự việc cụ thể hóa của khái niệm đó.

Trong một lớp có thể có nhiều sự thể hiện khác nhau.

#### Các thuộc tính (Properties)

Thuộc tính là sự mô tả các đối tượng trong Ontology.

Bao gồm 2 thành phần: Tên và Giá trị.

#### Các mối quan hệ (Relation)

Quan hệ giữa các đối tượng trong ontology cho biết đối tượng này liên hệ với đối tượng khác như thế nào.

Các loại quan hệ:

* Quan hệ giữa các lớp.
* Quan hệ giữa các thực thể.
  + Quan hệ giữa một thực thể và một lớp
  + Quan hệ giữa các cá thể và lớp.

### 2.2.3 Phân loại Ontology

Phân loại dựa trên phạm vi của các đối tượng được mô tả:

* Knowledge Representation ontology (ontology biểu diễn tri thức).

+Dựa trên các cách biểu diễn tri thức truyền thống.

+Ví dụ: Frame-Ontology.

* General/Common ontology (ontology tổng quát).

+ Từ vựng liên quan đến mọi thứ, sự kiện, thời gian, không gian, ...

+ Ví dụ: ontology về bảng trao đổi giữa meter và inch.

* Metadata-ontology (cung cấp từ vựng).

+Định nghĩa các ontology.

+Ví dụ: Registry Ontology, dùng để quản lý các ontology khác.

* Domain ontology (ontology lĩnh vực).

+Từ vựng của các khái niệm chỉ trong một phạm vi vừa và nhỏ.

+Ví dụ: ontology về lý thuyết hoặc các nguyên lý cơ bản của một miền.

* Task ontology (ontology tác vụ).

+Hệ thống các từ vựng có thể cùng hoặc không cùng phạm vi ứng dụng cụ thể.

* Domain – task ontology (ontology lĩnh vực – tác vụ).
* Application ontology (ontology ứng dụng).

### 2.2.4 Vai trò và lợi ích sử dụng Ontology

* Chia sẻ kiến thức chung giữa con người hoặc những tác tử phần mềm với nhau
* Có thể tái sử dụng kiến thức về một lĩnh vực.
* Làm rõ ràng những giả định thuộc chuyên ngành.
* Có thể phân tích và suy luận kiến thức chuyên ngành.
  + Ví dụ: Con người có thể đọc được dữ liệu với cấu trúc của XML dễ dàng.
* Biểu diễn tri thức phân tán giữa một số trang web.
* Cung cấp các thông tin giàu ngữ nghĩa.

### 2.2.5 Ontology và các phương pháp biểu diễn

**Bước 1: Xác định lĩnh vực và phạm vi của Ontology.**

Ontology miêu tả lĩnh vực nào?

Ontology phục vụ cho mục đích gì?

Cơ sở tri thức trong Ontology sẽ trả lời những câu hỏi gì?

Ontology đã có đủ thông tin để trả lời các câu hỏi được quan tâm trên cơ sở tri thức hay không?

**Bước 2: Xem xét việc sử dụng lại các ontology có sẵn.**

Giảm thiểu công sức xây dựng 1 ontology.

Khi tương tác giữa các ứng dụng với nhau, các ứng dụng cần hiểu các lớp, thực thể, quan hệ, … của nhau để trao đổi thông tin.

**Bước 3: Liệt kê các thuật ngữ quan trọng.**

Liệt kê các thuật ngữ xuất hiện trong miền quan tâm.

Các thuật ngữ là danh từ sẽ trở thành các lớp, danh từ sẽ trở thành thuộc tính, động từ là quan hệ giữa các lớp.

**Bước 4: Xác định các lớp và phân cấp của các lớp**

Định nghĩa các lớp từ một số thuật ngữ đã liệt kê ở bước 3

Xây dựng cấu trúc phân lớp, phân cấp theo quan hệ lớp cha-lớp con.

**Bước 5: Xác định các thuộc tính.**

Định nghĩa thuộc tính của lớp bên trong. Có 2 loại:

* Thuộc tính đơn: các giá trị đơn: chuỗi, số, …
* Thuộc tính phức: có thể tham khảo đến một đối tượng khác.

**Bước 6: Xác định ràng buộc của các thuộc tính.**

Ràng buộc quan trọng nhất đối với một thuộc tính:

* Lượng số: quy địnhsố giá trị mà 1 thuộc tính có thể nhận
* Kiểu: các kiểu mà thuộc tính có thể nhận là chuỗi, số, liệt kê, kiểu thực thế,

**Bước 7: Tạo các thể hiện / thực thể.**

Tạo thực thể cho mỗi lớp và gán giá trị cho các thuộc tính.

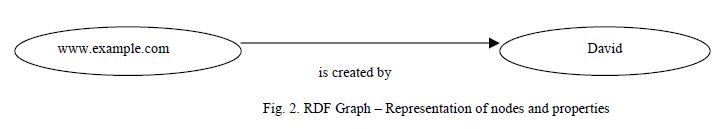
### 2.2.6 Các ngôn ngữ đặc tả Ontology

#### a. RDF

RDF (Resource Description Framework) làm cho thông tin có ngữ nghĩa như là những thông tin máy tính có thể truy cập.Nó là một mô hình chuẩn cho trao đổi dữ liệu trên web. RDF có nhiều tính năng để tạo điều kiện sát nhập dữ liệu ngay cả khi các lược đồ cơ bản khác nhau, và đặc biệt là hỗ trợ sự phát triển của lược đồ theo thời gian.

RDF mở rộng cấu trúc liên kết của trang web sử dụng URI để đặt tên cho mối quan hệ giữa các sự vật cũng như hai đầu của đường link (điều này thường được gọi là "triple"). Sử dụng mô hình đơn giản này, nó cho phép dữ liệu có cấu trúc và bán cấu trúc được pha trộn, tiếp xúc, và chia sẻ trên các ứng dụng khác nhau.

Cấu trúc liên kết này tạo thành một định hướng, gán nhãn đồ thị, khi đó các cạnh đại diện cho các liên kết có tên giữa hai nguồn tài nguyên (đại diện bởi các nút đồ thị). Biểu đồ này xem là mô hình đơn giản nhất có thể cho RDF ​​[3] và thường được sử dụng trong giải thích dễ hiểu trực quan.



*Hình 5: Ví dụ về một khái niệm biểu diễn bằng đồ thị RDF*

Các thành phần node và các thành phần thuộc tính (*Node elements and property elements)*

Một đồ thị RDF được đưa ra trong hình, nơi mà các nút được biểu diễn như là hình bầu dục biểu thị các lớp và cạnh đại diện cho các thuộc tính.Lớp và các thuộc tính có thể là một số tài nguyên khác.

**Hạn chế của sức mạnh ý nghĩa của RDF Schema**

RDF và RDFS cho phép các đại diện của một số kiến ​​thức ontology. Nguyên thủy mô hình chính của RDF / RDFS liên quan đến việc tổ chức từ vựng trong hệ thống phân cấp kiểu: phân lớp và thay thế các mối quan hệ thuộc tính, miền và giới hạn phạm vi, và các trường hợp của các lớp. Tuy nhiên một số tính năng khác bị mất. Đó là:

* Phạm vi cục bộ của thuộc tính: RDF: phạm vi biểu thị phạm vi của một thuộc tính, xem xét câu "Animal eat plants". Quy tắc này được áp dụng cho tất cả các loại động vật. Vì vậy, trong RDF Schema nó không thể áp dụng chỉ một giới hạn phạm vi đến các lớp nhất định. Ví dụ theo quy tắc trước nói rằng con bò chỉ ăn thực vật, trong khi các động vật khác có thể ăn thịt là một câu không hợp lệ.
* Tính tách biệt của các lớp: Đôi khi cần phải nói rằng các lớp thường tách rời. Ví dụ, nam và nữ thường tách rời. Nhưng trong RDF Schema được sử dụng để đại diện cho mối quan hệ phân lớp ví dụ nữ là lớp con của người.
* Kết hợp logic của các lớp: Đôi khi có một nhu cầu để xây dựng lớp mới bằng cách kết hợp các lớp khác sử dụng phép hợp, phép giao và phần bù. Ví dụ, biểu thị người lớp để được hợp của các lớp tách rời là nam và nữ. RDF Schema không cho phép định nghĩa như vậy.
* Hạn chế cardinality: Hạn chế về bao nhiêu giá trị khác nhau một thuộc tính có thể hoặc phải được biểu diễn như là tỷ lệ cardinality. Ví dụ, một người có đúng hai cha mẹ, và một khóa học được giảng dạy bởi ít nhất một giảng viên. Một lần nữa, những hạn chế đó là không thể thể hiện trong RDF Schema.
* Đặc điểm của thuộc tính: Đôi khi nó rất hữu ích để cho rằng một thuộc tính là bắc cầu (như "greater than"), độc đáo (như "is mother of"), hoặc nghịch đảo của thuộc tính khác (như "eats" và "is eaten by"). RDF không cho phép xác định các loại tính năng.

#### b. OWL

OWL (The Web Ontology Language) là một ngôn ngữ gần như XML dùng để mô tả các hệ cơ sở tri thức.OWL là một ngôn ngữ đánh dấu dùng để xuất bản và chia sẻ dữ liệu trên Internet thông qua những mô hình dữ liệu gọi là “ontology”.Ontology mô tả một lĩnh vực (domain) và diễn tả những đối tượng trong lĩnh vực đó cùng những mối quan hệ giữa các đối tượng này. OWL là phần mở rộng về từ vựng của RDF và được kế thừa từ ngôn ngữ DAML+OIL Web ontology – một dự án được hỗ trợ bởi W3C. OWL biểu diễn ý nghĩa của các thuật ngữ trong các từ vựng và mối liên hệ giữa các thuật ngữ này để đảm bảo phù hợp với quá trình xử lý bởi các phần mềm.

OWL được xem như là một kỹ thuật trọng yếu để cài đặt cho Semantic Web trong tương lai. OWL được thiết kế đặc biệt để cung cấp một cách thức thông dụng trong việc xử lý nội dung thông tin của Web. Ngôn ngữ này được kỳ vọng rằng sẽ cho phép các hệ thống máy tính có thể đọc được thay thế cho con người. Vì OWL được viết bởi XML, các thông tin OWL có thể dễ dàng trao đổi giữa các kiểu hệ thống máy tính khác nhau, sử dụng các hệ điều hành và các ngôn ngữ ứng dụng khác nhau.Mục đích chính của OWL là sẽ cung cấp các chuẩn để tạo ra một nền tảng để quản lý tài sản, tích hợp mức doanh nghiệp và để chia sẻ cũng như tái sử dụng dữ liệu trên Web. OWL được phát triển bởi nó có nhiều tiện lợi để biểu diễn ý nghĩa và ngữ nghĩa hơn so với XML, RDF và RDFS, và vì OWL ra đời sau các ngôn ngữ này, nó có khả năng biểu diễn các nội dung mà máy có thể biểu diễn được trên Web.

Các phiên bản của OWL: Hiện nay có ba loại OWL : OWL Lite, OWL DL (description logic), và OWL Full.

* *OWL Lite*: hỗ trợ cho những người dùng chủ yếu cần sự phân lớp theo thứ bậc và các ràng buộc đơn giản. Ví dụ: Trong khi nó hỗ trợ các ràng buộc về tập hợp, nó chỉ cho phép tập hợp giá trị của 0 hay 1. Điều này cho phép cung cấp các công cụ hỗ trợ OWL Lite dễ dàng hơn so với các bản khác.
* *OWL DL (OWL Description Logic*): hỗ trợ cho những người dùng cần sự diễn cảm tối đa trong khi cần duy trì tính tính toán toàn vẹn (tất cả các kết luận phải được đảm bảo để tính toán) và tính quyết định (tất cả các tính toán sẽ kết thúc trong khoảng thời gian hạn chế). OWL DL bao gồm tất cả các cấu trúc của ngôn ngữ OWL, nhưng chúng chỉ có thể được sử dụng với những hạn chế nào đó (Ví dụ: Trong khi một lớp có thể là một lớp con của rất nhiều lớp, một lớp không thể là một thể hiện của một lớp khác). OWL DL cũng được chỉ định theo sự tương ứng với logic mô tả, một lĩnh vực nghiên cứu trong logic đã tạo nên sự thiết lập chính thức của OWL.
* *OWL Full* muốn đề cập tới những người dùng cần sự diễn cảm tối đa và sự tự do của RDF mà không cần đảm bảo sự tính toán của các biểu thức. Ví dụ, trong OWL Full, một lớp có thể được xem xét đồng thời như là một tập của các cá thể và như là một cá thể trong chính bản thân nó. OWL Full cho phép một ontology gia cố thêm ý nghiã của các từ vựng được định nghĩa trước (RDF hoặc OWL).
* Các phiên bản này tách biệt về các tiện ích khác nhau, OWL Lite là phiên bản dễ hiểu nhất và phức tạp nhất là OWL Full.

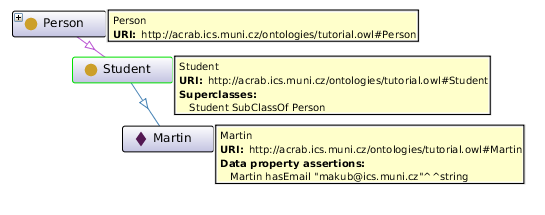
**Mối liên hệ giữa các ngôn ngữ con của OWL:**

* Mọi ontology hợp lệ dựa trên OWL Lite đều là ontology hợp lệ trên OWL DL.
* Mọi ontology hợp lệ dựa trên OWL DL đều là ontology hợp lệ trên OWL Full.
* Mọi kết luận hợp lệ dựa trên OWL Lite đều là kết luận hợp lệ trên OWL DL.
* Mọi kết luận hợp lệ dựa trên OWL DL đều là kết luận hợp lệ trên OWL Full.

2.3 Mô hình OWL2 trong Protégé

1. OWL Ontology

* OWL **ontology** là một tập **các tiên đề**, cung cấp các khẳng định logic tường minh về ba loại – **lớp**, **thể hiện** và **thuộc tính**. Bằng cách sử dụng một phần mềm gọi là **bộ suy diễn** (reasoner) ta có thể suy ra các sự kiện ngầm chứa trong mô hình ontology. Ví dụ nếu một thể hiện Martin thuộc lớp Student, và lớp Student là lớp con của lớp Person, một bộ suy diễn sẽ suy ra rằng Martin là một Person.
* Có hai loại thuộc tính trong OWL ontology. **Thuộc tính dữ liệu** là các quan hệ nhị phân liên kết một thể hiện đến một giá trị có kiểu, như xsd:dateTime hoặc xsd:string literal. **Thuộc tính đối tượng** là các quan hệ nhị phân liên kết một đối tượng đến một đối tượng.
* Có nhiều cách làm một OWL ontology có thể *được viết*, tuy nhiên điều quan trọng là *ngữ nghĩa*, được thể hiện ở hình ảnh Ontograf dưới đây, được xây dựng bởi công cụ Protégé. Đây là một ontology nhỏ với hai lớp và một thể hiện, trong đó có một thuộc tính dữ liệu:



* Các tiên đề tương tự có thể được viết bằng nhiều cách. Phổ biết nhất là định dạng XML/RDF, đặc tả OWL 2 sử dụng định dạng *functional syntax*, các tác giả Protégé phát triển định dạng ngắn hơn Manchester, và định dạng ngắn gọn nhất là Turtle.

1. Các loại tiên đề

* Có rất nhiều loại tiên đề có thể được biểu diễn trong OWL.
  1. Khai báo
* **Khai báo lớp** định nghĩa một lớp. Một lớp có thể chứa nhiều thể hiện.

Class declaration

Declaration( Class( :Person ) )

* **Khai báo thể hiện** định nghĩa một thể hiện có tên.

Named individual declaration

Declaration( NamedIndividual( :Martin ) )

* **Khẳng định lớp** phát biểu một thể hiện thuộc về một lớp.

Class assertion

ClassAssertion( :Student :Martin )

* **Khẳng định lớp con** phát biểu tất cả thể hiện thuộc một lớp cũng thuộc một lớp khác.

Subclass

SubClassOf( :Student :Person )

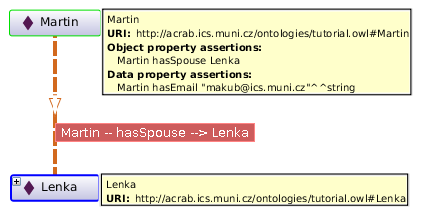
* **Khai báo thuộc tính** định nghĩa hoặc thuộc tính dữ liệu kiến kết một thể hiện đến dữ liệu, hoặc thuộc tính đối tượng liên kết đến một thể hiện.



Declaration( DataProperty( :hasEmail ) )

Declaration( ObjectProperty( :hasSpouse ) )

* **Khẳng định thuộc tính** phát biểu quan hệ giữa một thể hiện đến một dữ liệu hoặc thể hiện.



DataPropertyAssertion( :hasEmail :Martin "makub@ics.muni.cz"^^xsd:string )

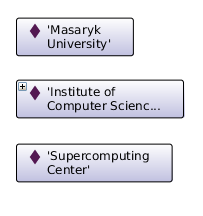
ObjectPropertyAssertion( :hasSpouse :Martin :Lenka )

* **Khẳng định thuộc tính tiêu cực** phát biểu quan hệ giữa một thể hiện đến một dữ liệu hoặc thể hiện **không tồn tại**. OWL sử dụng Open World Assumption, nên nếu một thể hiện không được liên kết bởi những thuộc tính có những giá trị, nó có thể bị gây ra bởi hai lý do – hoặc nó thực sự không có thuộc tính có giá trị, hoặc nó **không xác định** bởi thông tin còn thiếu từ ontology. Một khẳng định thuộc tính tiêu cực phát biểu rằng thể hiện không thể có giá trị thuộc tính đó.

NegativeObjectPropertyAssertion( :hasSpouse :Martin :Peter)

NegativeDataPropertyAssertion( :hasEmail :Martin "president@whitehouse.gov")

* **Khẳng định chú thích** cho phép chú thích bất kỳ thứ gì với một số thông tin, ví dụ ta có thể dùng ICS như là tên của một thể hiện bởi vì nó là tên viết tắt, và sử dụng chú thích để gán nhãn tên viết tắt với ý nghĩa đầy đủ “Institute of Computer Science”.



Declaration( NamedIndividual( :MU ))

AnnotationAssertion( rdfs:label :MU "Masaryk University" )

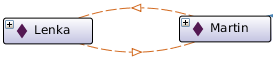
Declaration( NamedIndividual( :ICS ) )

AnnotationAssertion( rdfs:label :ICS "Institute of Computer Science" )

Declaration( NamedIndividual(:SC) )

AnnotationAssertion( rdfs:label :SC "Supercomputing Center" )

* 1. Tiên đề thuộc tính
* Ta có thể định nghĩa nhiều thông tin về các thuộc tính, một thuộc tính là **transitive**, **symmetic**, **asymmetric**, **reflexive**, **irreflexive**, **functional** (có thể chỉ có một giá trị), **inverse-functional** (nghịch đảo của nó là functional), **inverse** đến các thuộc tính khác, **subproperty** của các thuộc tính khác, **equivalent** đến các thuộc tính khác, hoặc **disjoint** với các thuộc tính khác (hai thể hiện có thể không được liên kết bởi cả hai thuộc tính trong cùng một thời điểm).
* Trong một ví dụ đơn giản, ta có thể định nghĩa một thuộc tính *hasSpouse* là **symmetric**, **functional** và **irreflexive**:

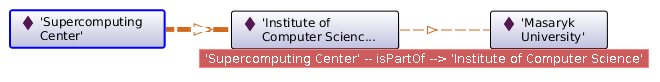


SymmetricObjectProperty( :hasSpouse )

FunctionalObjectProperty( :hasSpouse )

IrreflexiveObjectProperty( :hasSpouse )

* Ở một ví dụ phức tạp hơn, một thuộc tính **transitive** *isPartOf* được định nghĩa, kết nối những bộ phận tổ chức của Đại học Masaryk. Sau đó thuộc tính *hasPart* được định nghĩa như một thuộc tính **inverse** của *isPartOf*.



Declaration( ObjectProperty( :isPartOf ))

TransitiveObjectProperty( :isPartOf )

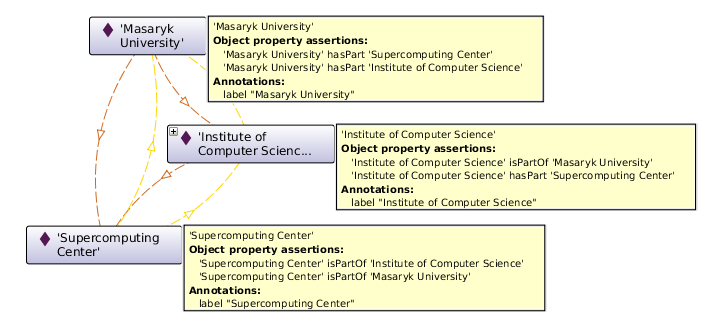
ObjectPropertyAssertion( :isPartOf :SC :ICS )

ObjectPropertyAssertion( :isPartOf :ICS :MU )

Declaration( ObjectProperty(:hasPart) )

InverseObjectProperties( :hasPart :isPartOf )

* Giờ ta có thể sử dụng một **bộ suy diễn** để suy ra các thuộc tính khác. Ví isPartOf là transitive, SCB isPartOf MU, và vì hasPart là nghịch đảo của isPartOf, ta có được các quan hệ được suy ra hoàn chỉnh.



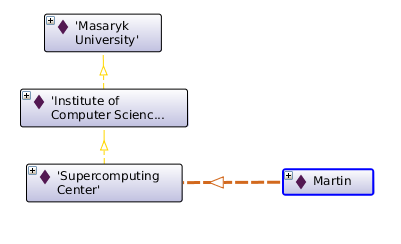
ObjectPropertyAssertion( :isPartOf :SC :MU )

ObjectPropertyAssertion( :hasPart :MU :SC )

ObjectPropertyAssertion( :hasPart :MU :ICS )

ObjectPropertyAssertion( :hasPart :ICS :SC )

* Nó cho thấy sức mạnh của suy diễn – ta không cần phải kháo báo tất cả quan hệ, chỉ cần những quan hệ tối thiểu, và các quan hệ còn lại có thể được suy ra.
  1. Mắt xích thuộc tính
* **Mắt xích thuộc tính** cho phép định nghĩa những quan hệ giữa ba thể hiện, ví dụ nổi bật nhất là thuộc tính *uncle* được định nghĩa như một mắt xích của thuộc tính *parent* và *brother*.
* Một thuộc tính có thể mắt xích với chính nó. Ví dụ, ta định nghĩa thuộc tính *isEmployedAt*, là một mắt xích của chính nó và thuộc tính transitive *isPartOf*, nghĩa là nếu một người đi làm ở những đơn vị tổ chức thì người đó cũng làm ở các đơn vị tổ chức lớn hơn.

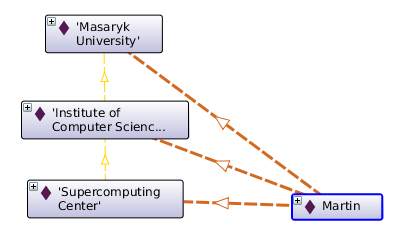


Declaration( ObjectProperty( :isEmployedAt ) )

ObjectPropertyAssertion( :isEmployedAt :Martin :SC )

SubObjectPropertyOf( ObjectPropertyChain( :isEmployedAt :isPartOf ) :isEmployedAt)

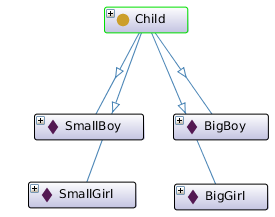
* Một lần nữa, ta sử dụng bộ suy diễn để suy ra các quan hệ hoàn chỉnh.



ObjectPropertyAssertion( :isEmployedAt :Martin :ICS )

ObjectPropertyAssertion( :isEmployedAt :Martin :MU )

* 1. Biểu thức lớp
* Lớp có thể được định nghĩa bằng **biểu thức lớp**. Một định nghĩa lớp thông thường là một khẳng định một lớp có tên **tương đương** với những lớp (không tên/vô danh) được định nghĩa bằng một biểu thức. Lớp cũng có thể được định nghĩa là **disjoined** với lớp khác, nghĩa là chúng không chia sẻ thể hiện nào với nhau.
* **Các phép trong tập (set): liệt kê, hợp, giao, hội**
* Ví dụ, ta định nghĩa một lớp mới *Child* chứa bốn thể hiện mới có tên *SmallBoy*, *BigBoy*, *SmallGirl*, *BigGirl*. Những thể hiện này phải được khai báo khác nhau, nếu không một bộ suy diễn OWL có thể xem chúng là như nhau.



Declaration( Class( :Child ) )

Declaration( NamedIndividual( :BigBoy ) )

Declaration( NamedIndividual( :BigGirl ) )

Declaration( NamedIndividual( :SmallBoy ) )

Declaration( NamedIndividual( :SmallGirl ) )

ClassAssertion( :Child :BigBoy )

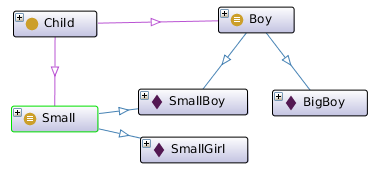
ClassAssertion( :Child :BigGirl )

ClassAssertion( :Child :SmallBoy )

ClassAssertion( :Child :SmallGirl )

DifferentIndividuals( :BigBoy :BigGirl :SmallBoy :SmallGirl )

* + 1. Phép liệt kê
* Ta định nghĩa hai lớp cơ sở bằng cách liệt kê các thành viên, lớp *Boy* chứa những con trai và lớp *Small* chứa những đứa trẻ nhỏ con. Ví dụ, thể hiện SmallBoy vừa trong lớp Boy và lớp Small.



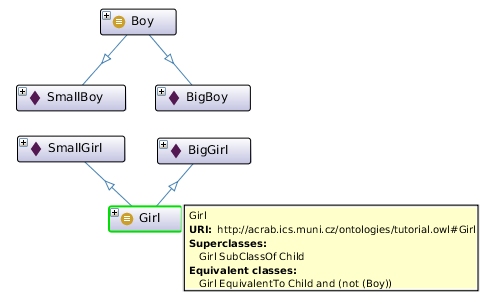
SubClassOf(:Boy :Child)

SubClassOf(:Small :Child)

EquivalentClasses(:Boy ObjectOneOf(:SmallBoy :BigBoy))

EquivalentClasses(:Small ObjectOneOf(:SmallBoy :SmallGirl))

* + 1. Phép hội và giao
* Giờ ta định nghĩa lớp mới *Girl* là lớp **giao** giữa lớp *Child* và lớp không tên là **hội** của lớp *Boy*. Lớp *Girl* tương đương với **Child and not Boy**, nghĩa là một con gái là một đứa trẻ không phải là một con trai.
* Bộ suy diễn sẽ tìm ra rằng lớp Girl chứa những thể hiện SmallGirl và BigGirl.

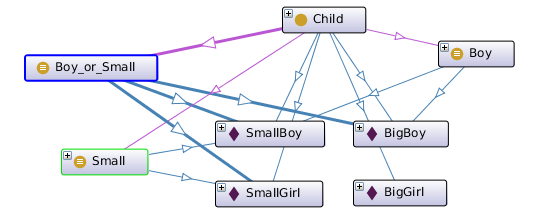


Declaration( Class( :Girl ) )

SubClassOf( :Girl :Child )

EquivalentClasses( :Girl ObjectIntersectionOf( ObjectComplementOf( :Boy ) :Child ) )

* + 1. Phép hợp
* Ta có thể định nghĩa một lớp mới *Boy\_or\_small* là **hợp** giữa lớp *Small* và *Boy*, nghĩa là **Boy or Small**. Bộ suy diễn sẽ tìm ra rằng lớp chứa những thể hiện *SmallBoy*, *SmallGirl*, *BigBoy*.



Declaration( Class( :Boy\_or\_Small ) )

EquivalentClasses( :Boy\_or\_Small ObjectUnionOf( :Small :Boy ) )

* 1. Định lượng hiện sinh (Existential quantification)
* Một biểu thức lớp có thể nói rằng lớp chỉ chứa những thể hiện được kết nối bởi một thuộc tính cho trước với những thể hiện từ một lớp cho trước. Ta có thể định nghĩa một lớp mới OrgUnit chứa ba thể hiện MU, ICS, SC.

Declaration( Class( :OrgUnit ))

ClassAssertion( :OrgUnit :ICS )

ClassAssertion( :OrgUnit :MU )

ClassAssertion( :OrgUnit :SC )

* Sau đó ta có thể định nghĩa một lớp mới *Employee* là lớp con của *Person*, nghĩa là **tương đương** với một lớp vô danh của những thể hiện được liên kết bởi thuộc tính *isEmployedAt* đến những thể hiện từ lớp *OrgUnit*. Nghĩa là những nhân viên là những người được tuyển dụng ở đâu đó.

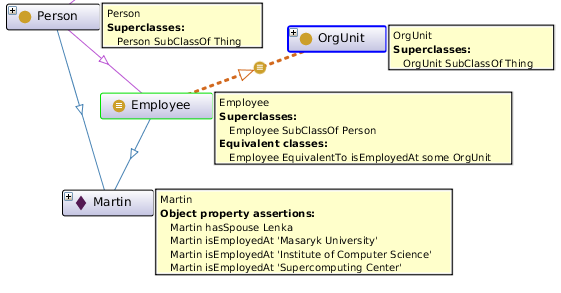


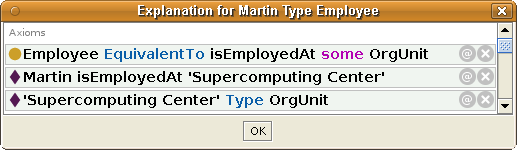
Declaration( Class( :Employee) )

SubClassOf( :Employee :Person )

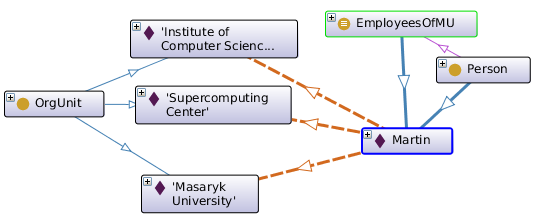
EquivalentClasses( :Employee ObjectSomeValuesFrom( :isEmployedAt :OrgUnit ) )

* Bằng sử dụng một bộ suy diễn ta có thể suy ra rằng lớp Employee chứa thể hiện Martin.





* 1. Ràng buộc giá trị thể hiện
* Một biểu thức lớp cũng có thể nói rằng lớp chứa **những thể hiện được kết nối bởi một thuộc tính cho trước với một thể hiện cho trước**.
* Ví dụ, ta định nghĩa một lớp mới *EmployeesOfMU* chứa những thể hiện được kết nối bởi thuộc tính *isEmployedAt* với thể hiện *MU*, nghĩa là những người làm việc tại MU. Bộ suy diễn sẽ tìm thấy lớp chứa thể hiện *Martin*.

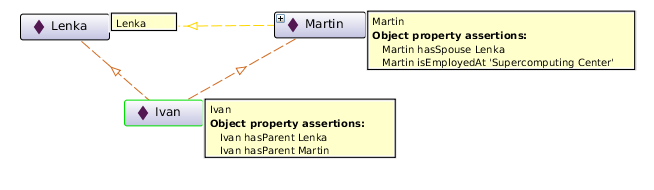


Declaration( Class( :EmployeesOfMU ) )

SubClassOf( :EmployeesOfMU :Person )

EquivalentClasses( :EmployeesOfMU ObjectHasValue( :isEmployedAt :MU ) )

* 1. Định lượng phổ quát (Universal quantification)
* Định lượng phổ quát (phép ∀ trong toán học) trong OWL có một chút khác biệt bởi tính **open world assumption** mà OWL có.
* Một định lượng hiện sinh được mô tả ở trên (Phép của OWL) được định nghĩa trong OWL 2 Direct Semantics phát biểu “những thể hiện mà *có sự tồn tại* thể hiện từ một lớp cho trước được kết nối bới một thuộc tính cho trước”, trong ký hiệu toán học ObjectSomeValuesFrom( OPE CE ) : { x | ∃ y : ( x, y ) ∈ (OPE)OP and y ∈ (CE)C } với OPE biểu thị một biểu thức thuộc tính đối tượng, .OP là hàm thuộc tính đối tượng diễn dịch (interpertation), CE biểu thị một biểu thức lớp, .C là hàm lớp diễn dịch.
* Định lượng phổ quát được định nghĩa là phép ObjectAllValuesFrom( OPE CE ) : { x | ∀ y : (x, y) ∈ (OPE)OP với y ∈ (CE)C } nghĩa là **những thể hiện mà luôn được kết nối bởi thuộc tính cho trước chỉ đến những thể hiện từ lớp cho trước**.
* Vấn đề với open world assumption là trong một thế giới mở một ontology có thể không chứa tất cả những thông tin. Ví dụ, có một ontology phát biểu rằng thể hiện Peter có hai đứa con, con trai John và Paul. Ta có thể định nghĩa một lớp ManWhoHaveOnlySons là lớp của các thể hiện được kết nối bởi thuộc tính *hasChild* chỉ đến những thể hiện từ lớp *Boy*, nghĩa là ObjectAllValuesFrom( :hasChild :Boy). Tuy nhiên thể hiện Peter không nhất thết trong một lớp, vì Peter cũng có thể có một đứa con gái Jane, chỉ là ontology không chưa thông tin này.
* Định lượng phổ quát trong OWL chỉ hoạt động với **giới hạn số lượng** (cardinality restrictions), nên nếu ta biết rằng một thể hiện có thể được kết nối bởi một lớp cho trước đến một số **tối đa** các thể hiện khác nhau, và ta biết toàn bộ thể hiện, ta có thể rút ra được kết luận.
* Ví dụ, mỗi người có tối đa hai cha mẹ (parents). Nếu cả hai cha mẹ đều có cùng quốc tịch, thì người đó cũng có cùng quốc tịch.
* Ta tạo một thể hiện mới *Ivan*, được kết nối bởi một thuộc tính đối tượng mới *hasParent* đến những thể hiện *Martin* và *Lenka*. Nghĩa là Ivan là một đứa con của Martin và Lenka.



Declaration( NamedIndividual( :Ivan ) )

ClassAssertion( :Person :Ivan )

Declaration( ObjectProperty( :hasParent ) )

ObjectPropertyAssertion( :hasParent :Ivan :Martin )

ObjectPropertyAssertion( :hasParent :Ivan :Lenka )

* Ta có thể nói rằng một Person có tối đa hai cha mẹ, và định nghĩa một lớp mới *Czech* chứa *Martin* và *Lenka*. Điều quan trọng mới là ở hai tiên đề cuối cùng – định lượng phổ quát rằng các thể hiện có tất cả các cha mẹ Czech là (một lớp con của) người Czech, và mọi thể hiện có tên của ontology là khác nhau.

SubClassOf( :Person ObjectMaxCardinality( 2 :hasParent ) )

Declaration( Class( :Czech ) )

SubClassOf( :Czech :Person )

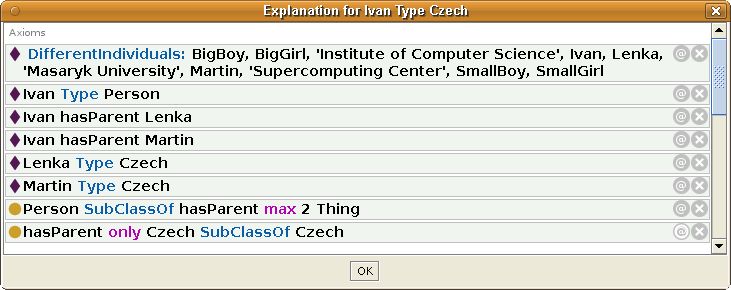
ClassAssertion( :Czech :Lenka )

ClassAssertion( :Czech :Martin )

SubClassOf( ObjectAllValuesFrom( :hasParent :Czech ) :Czech)

DifferentIndividuals( :BigBoy :BigGirl :ICS :Ivan :Lenka :MU :Martin :SC :SmallBoy :SmallGirl)

* Với những tiên đề này, ta có một kết luận rút ra được từ bộ suy diễn là – **Ivan thuộc lớp Czech**. Anh ấy là một Person, nên anh ấy có thể có tối đa hai cha mẹ, anh ấy có hai cha mẹ là hai thể hiện khác nhau, và đều là người Czech. Do đó không có thêm cha mẹ nào tồn tại cho anh, và định lượng phổ quát được thỏa mãn – anh ấy chỉ có cha mẹ người Czech.

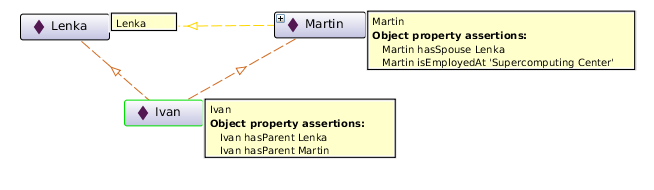


* 1. Ràng buộc kiểu dữ liệu
* Đối với các thuộc tính dữ liệu, ta có thể ràng buộc các giá trị của một thuộc tính dữ liệu. Ví dụ như tiên đề sau định nghĩa rằng những thể hiện của lớp Person có đúng một giá trị của thuộc tính hasAge là một số nguyên trong khoảng từ 0 đến 130.

EquivalentClasses( :Person DataExactCardinality( 1 :hasAge DatatypeRestriction( xsd:integer xsd:minInclusive "0"^^xsd:integer xsd:maxInclusive "130"^^xsd:integer)))

1. Luật SWRL

* Ngôn ngữ OWL 2 không thể thể hiện tất cả các quan hệ. Ví dụ như không thể biểu diễn quan hệ *child of married parents*, vì không thể trong OWL 2 để diễn tả quan hệ giữa các thể hiện nà một thể hiện có những quan hệ tới.
* Cách biểu diễn của OWL có thể được mở rộng bằng luật SWRL (Semantic Web Rule Language) cho một ontology. Các luật SWRL tương tự như ngôn ngữ Prolog hay DATALOG. Thực tế, luật SWRL là luật DATALOG với unary predicates để mô tả lớp và kiểu dữ liệu, binary predicates cho thuộc tính, và những predicates built-in n-ary đặc biệt.
* Với ví dụ về Ivan, đứa con của Martin và Lenka, thuộc tính symmetric *hasSpouse* kết nối Martin và Lenka.



SymmetricObjectProperty(:hasSpouse)

ObjectPropertyAssertion(:hasSpouse :Martin :Lenka)

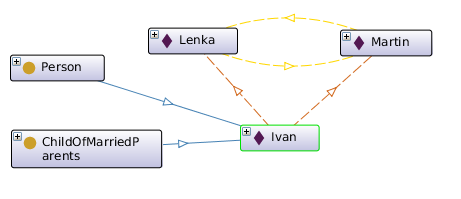
ObjectPropertyAssertion( :hasParent :Ivan :Martin )

ObjectPropertyAssertion( :hasParent :Ivan :Lenka )

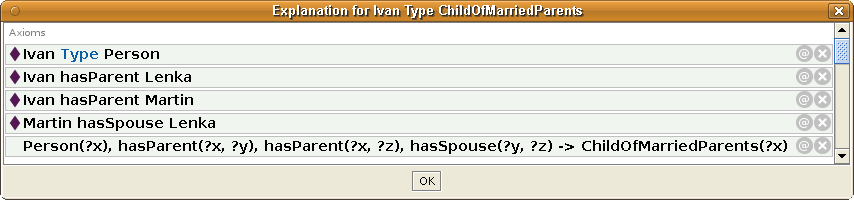
* Ta có thể thêm một luật SWRL nói rằng một thể hiện X trong lớp Person, mà có cha mẹ Y và Z với Y có vợ/chồng là Z, sẽ thuộc lớp mới *ChildOfMarriedParents*.

Person(?x), hasParent(?x, ?y), hasParent(?x, ?z), hasSpouse(?y, ?z) -> ChildOfMarriedParents(?x)

* Khi ta sử dụng bộ suy diễn có hỗ trợ luật SWRL, nó suy ra được *Ivan* thuộc lớp *ChildOfMarriedParents*.



ClassAssertion( :ChildOfMarriedParents :Ivan )



1. Thể hiện khác biệt và khóa

* **Open world assumption** của OWL nói rằng một ontology có thể không chứa tất cả mọi dữ liệu và do đó vài thông tin có thể là không xác định. Đây là lý do vì sao **những thể hiện có tên khác nhau có thể đại diện cho cùng thể hiện** trì khi nó được tường minh phát biểu rằng chúng khác nhau. Cũng là lý do vì sau ta phải chỉ định ở những ví dụ bên trên (các phép trong tập và định lượng phổ quát) rằng những thể hiện là khác nhau.
* Ta có thể khai báo các thể hiện có tên khác nhau bằng hai cách. Cách một là bằng tiên đề DifferentIndividuals()

DifferentIndividuals(:BigBoy :BigGirl :SmallBoy :SmallGirl :ICS :MU :SC :Ivan :Lenka :Martin )

* Tuy nhiên cách này được cho là gây vấn đề về hiệu năng cho bộ suy diễn khi số lượng thể hiện lớn. Cách thứ hay để chỉ những thể hiện khác nhau là bằng thuộc tính dữ liệu function với những giá trị duy nhất. Thuộc tính functional chỉ có thể có một giá trị, do dó làm cho các thể hiện **khác nhau** khi giá trị của chúng khác nhau. Nếu thuộc tính được khai báo là một **khóa**, nó làm cho các thể hiện có **cùng** giá trị là cùng một thể hiện.

Declaration(DataProperty(:hasId))

FunctionalDataProperty(:hasId)

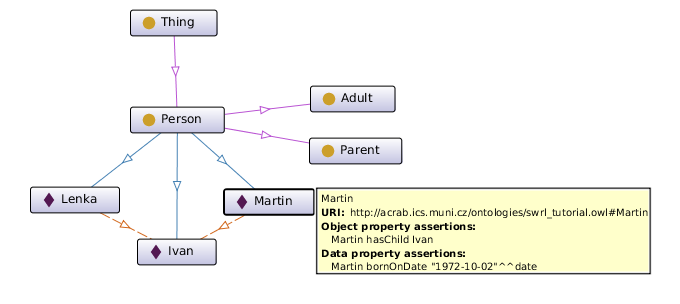
HasKey( :Person () (:hasId) )

DataPropertyAssertion(:hasId :Lenka "1234")

DataPropertyAssertion(:hasId :Martin "5648")

1. SWRL predicates

* Luật SWRL có thể sử dụng những predicate khác hơn là chỉ với tên lớp và tên thuộc tính. Những predicate có thể là
  1. **Biểu thức lớp** – biểu thức lớp tùy ý, không chỉ với lớp có tên
  2. **Biểu thức thuộc tính** – hiện không có nhu cầu sử dụng biểu thức thuộc tính trong SWRL
  3. **Ràng buộc miền dữ liệu** – chỉ định kiểu cho giá trị dữ liệu, như số nguyên, ngày, hợp của các kiểu XML Schema, kiểu liệt kê
  4. **sameIndividual** và **differentIndividuals** – để xác định thể hiện giống và khác nhau
  5. **core SWRL built-ins** – những predicate đặc biệt được định nghĩa trong SWRL proposal để thao tác với các giá trị dữ liệu, ví dụ như cộng thêm một số
  6. **custom SWRL buil-ins** – ta có thể tự định nghĩa các built-ins bằng code
* Ví dụ, một ontology đơn giản có luật, nó định nghĩa ba thể hiện với ngày sinh:



Declaration(Class(:Person))

Declaration(ObjectProperty(:hasChild))

Declaration(Class(:Parent))

Declaration(Class(:Adult))

SubClassOf(:Adult :Person)

SubClassOf(:Parent :Person)

Declaration(DataProperty(:bornOnDate))

Declaration(DataProperty(:bornInYear))

Declaration(DataProperty(:hasAge))

FunctionalDataProperty(:hasAge)

Declaration(DataProperty(:hasDriverAge))

Declaration(NamedIndividual(:Martin))

ClassAssertion(:Person :Martin)

DataPropertyAssertion(:bornOnDate :Martin "1972-10-02"^^xsd:date)

Declaration(NamedIndividual(:Lenka))

ClassAssertion(:Person :Lenka)

DataPropertyAssertion(:bornOnDate :Lenka "1975-11-10"^^xsd:date)

Declaration(NamedIndividual(:Ivan))

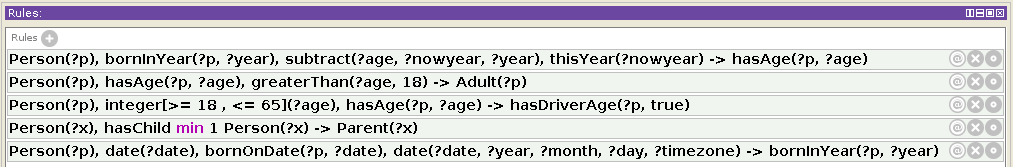
ClassAssertion(:Person :Ivan)

DataPropertyAssertion(:bornOnDate :Ivan "2006-04-14"^^xsd:date)

ObjectPropertyAssertion(:hasChild :Martin :Ivan)

ObjectPropertyAssertion(:hasChild :Lenka :Ivan)

* Ontology cũng định nghĩa một vài luật:

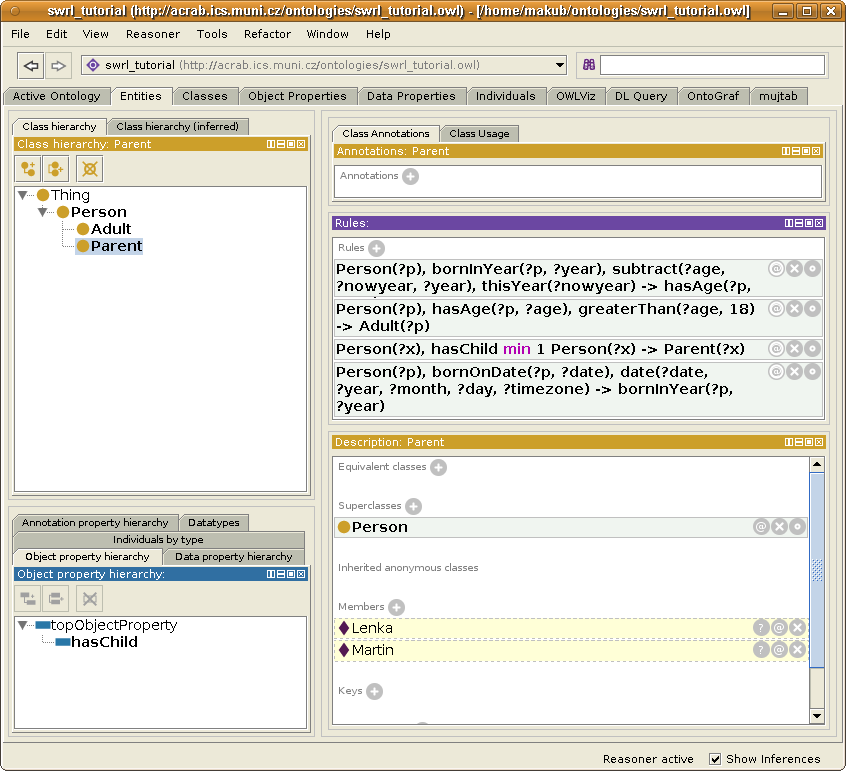


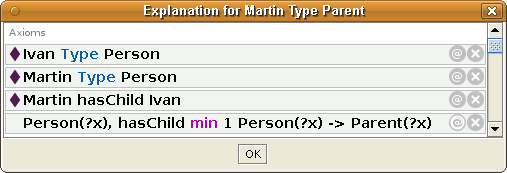
1. Luật SWRL với biểu thức lớp

* Luật ở danh sách trên được biểu diễn như

Person(?x), hasChild min 1 Person(?x) -> Parent(?x)

* Nghĩa là với những thể hiện, mà thuộc lớp Person, và có ít nhất một thuộc tính hasChild với một thể hiện thuộc lớp Person, phải thuộc lớp Parent. Hay nói cách khác, những người đó với có ít nhất một đứa con là cha mẹ.
* Khi sử dụng bộ suy diễn, nó sẽ suy ra rằng Martin và Lenka ở cùng lớp Parent:





* Cũng trong ví dụ này, ta có thể mô hình hóa lại đơn giản bằng thiết lập lớp Parent là lớp con của biểu thức lớp “(hasChild min 1 Person)” trong OWL. Hoặc là nó tương đương với biểu thức lớp, là một khẳng định cao hơn. Luật SWRL chỉ thực sự cần cho mô hình cấu trúc không phải dạng cây, không thể mô hình trong OWL.

1. Luật SWRL với ràng buộc miền dữ liệu

* Luật sau sử dụng ràng buộc miền dữ liệu:

Person(?p), integer[>= 18 , <= 65](?age), hasAge(?p, ?age) -> hasDriverAge(?p, true)

* Ràng buộc thỏa mãn khi biến ?age có giá trị số nguyên giữa 18 và 65.
* Ví dụ này biểu diễn một ràng buộc miền dữ liệu phức tạp với facets, một ví dụ đơn giản sẽ là hoặc nhằm ràng buộc kiểu mà không xài facets.
* Một ví dụ miền dữ liệu sử dụng phép hội, hợp, giao và facets cùng một lúc:

((integer[>=1,<=2] or integer[>5,<7]) and not ({0}))(?y)

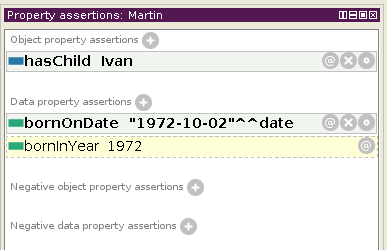
1. Luật SWRL với core built-ins

* SWRL proposal cũng định nghĩa những predicate đặc biệt, dùng để thao tác với các giá trị dữ liệu. Những predicate này được gọi là SWRL core built-ins, và được xác định bởi các IRI trong namespace swrlb.
* Những luật sau từ danh sách trên sử dụng core built-ins:

Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 18) -> Adult(?p)

Person(?p), bornOnDate(?p, ?date), xsd:date(?date), swrlb:date(?date, ?year, ?month, ?day, ?timezone) -> bornInYear(?p, ?year)

* Luật thứ nhất nói rằng những người có tuổi lớn hơn 18 là người lớn. Luật thứ hai lấy giá trị dữ liệu từ thuộc tính bornOnDate, phải có kiểu xsd:date, phân nhỏ nó thành nhiều phần, và thêm phần năm thành giá trị của thuộc tính bornInYear cho người đó.



1. Luật SWRL với custom built-ins

* Ta có thể tự định nghĩa SWRL built-ins.
* Luật sau sử dụng my:this year custom-built-in để bind giá trị số nguyên của năm hiện tại cho biến ?nowyear, và dùng swrlb:subtract core built-in để tính tuổi theo năm, và gán kết quá thành giá trị của thuộc tính hasAge:

Person(?p), bornInYear(?p, ?year), my:thisYear(?nowyear), swrlb:subtract(?age, ?nowyear, ?year) -> hasAge(?p, ?age)

1. Giới hạn của OWL 2 và SWRL

* Propositional logic
* SWRL chỉ là một mảnh của first order predicate logic, SWRL không thể biểu diễn những loại sau:
  1. Biểu thức mờ - “It often rains in autumn.”
  2. non-monotonicity - “Birds fly, penguin is a bird, but penguin does not fly.”
  3. propositional attitudes - “Eve thinks that 2 is not a prime number.” (It is true that she thinks it, but what she thinks is not true.)
  4. modal logic
     + possibility and necessity - “It is possible that it will rain today.”
     + epistemic modalities - “Eve knows that 2 is a prime number.”
     + temporal logic - “I am always hungry.”
     + deontic logic - “You must do this.”

c. JSON-LD

Để lưu trữ dữ liệu OWL, nhóm sử dụng ngôn ngữ JSON để mã hóa Dữ liệu Liên kết (Linked Data) với JSON-LD.

1. Linked Data nguồn https://en.wikipedia.org/wiki/Linked\_data

* Dữ liệu Liên kết là phương pháp đưa dữ liệu có cấu trúc liên kết lên Web và được truy vấn bằng ngữ nghĩa (semantic queries). Dữ liệu Liên kết xây dựng dựa trên chuẩn công nghệ Web như HTTP, RDF và URI, nhưng thay vì sử dụng LD để cung cấp trang Web cho cho người dùng như các công nghệ Web này, nó được mở rộng hơn ở chỗ cho phép chia sẽ dữ liệu tự động giữa các máy tính với nhau. Điều này giúp các dữ liệu từ các nguồn khác nhau được hợp nhất lại và được truy vấn chung.
* Tim Berners-Lee, giám đốc Hiệp hội World Wide Web, nêu ra bốn nguyên tắc cho Dữ liệu Kiên Kết:
  1. Sử dụng URI để đặt tên định danh vật.
  2. Sử dụng HTTP URI để các vật có thể được tìm kiếm.
  3. Cung cấp những thông tin hữu ích về tên định danh vật cho quá trình tìm kiếm, sử dụng các chuẩn mở như RDF, SPARQL.
  4. Tham khảo (Refer) đến những vật khác thông qua HTTP URI của chúng trước khi đưa dữ liệu lên Web.

1. JSON-LD nguồn https://en.wikipedia.org/wiki/JSON-LD

* JSON-LD, hay **J**ava**S**cript **O**bject **N**otation for **L**inked **D**ata, là phương pháp mã hóa dữ liệu liên kết bằng JSON. Nó là chuẩn được khuyến nghị bởi Hiệp hội World Wide Web.
* JSON-LD được thiết kế dựa trên khái niệm về “bối cảnh” (context) để giúp ánh xạ từ JSON đến một mô hình RDF. Bối cảnh sẽ kết nối dữ liệu thuộc tính từ một văn bản JSON đến những khái niệm trong một ontology. Để ánh xạ cú pháp từ JSON-LD sang RDF, JSON-LD ràng buộc các giá trị đến một kiểu nhất định hoặc được đính kèm với một ngôn ngữ. Một bối cảnh có thể được nhúng trực tiếp trong một văn bản JSON-LD hoặc được chia nhỏ thành nhiều tập tin và được tham khảo từ các văn bản khác nhau.
* Ví dụ:

{

"@context": {

"name": "http://xmlns.com/foaf/0.1/name",

"homepage": {

"@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",

"@type": "@id"

},

"Person": "http://xmlns.com/foaf/0.1/Person"

},

"@id": "http://me.example.com",

"@type": "Person",

"name": "John Smith",

"homepage": "http://www.example.com/"

}

* Ví dụ trên mô tả một người dựa trên bộ từ điển FOAF. Đầu tiên, hai thuộc tính JSON *name* và *homepage* và kiểu *Person* được ánh xạ đến những khái niệm trong bộ từ điển FOAF và giá trị của thuộc tính *homepage* được xác định bởi kiểu *@id*, tức giá trị được xem là một IR, là một trong các định nghĩa của bối cảnh. Dựa trên mô hình RDF, điều này cho phép mô tả một người được xác định rõ ràng bằng một IRI. Việt sử dụng các IRI giúp người dùng sử dụng văn bản RDF có thể phát hiện các dữ liệu mới dựa trên các liên kết; nguyên tắc này gọi là Follow Your Nose.
* Khi các dữ liệu được chú thích một cách có ngữ nghĩa như ví dụ trên, một bộ xử lý RDF có thể xác định rằng văn bản có thông tin về một người (*@type*) và nếu bộ xử lý hiểu được bộ từ điển FOAF, nó có thể xác định tên và homepage của người đó.

**2.4 Hệ Luật dẫn**

Một hệ luật dẫn được mô tả dưới dạng nếu <giả thuyết> thì <kết luận>. Đây là dạng

biểu diễn tri thức cơ bản nhất và rất phổ biến. Mô hình biểu diễn dạng này thì thường bao gồm : tập mô tả các sự kiện (có cấu trúc đơn giản) và tập luật dẫn. Trong đó phần giả thuyết và kết luận là tập các sự kiện. Mỗi sự kiện được mô tả dưới dạng <Đối tượng> < thuộc tính giá trị>. Hệ luật dẫn là tri thức có dạng gồm 1 tập các luật dẫn trên các sự kiện hay các tác vụ.

Hệ luật dẫn gồm 2 thành phần chính: (F, R)

Trong đó:

* **F(Facts)**: tập các sựkiện hay tác vụ
* **R(Rules)**: tập luật dẫn, mỗi luật có dạng r: gt(r) => kl(r)
* **p1, p2, …, pm -> q1,q2,….,qn**: trong đó pi, qjđều thuộc F

Thường tổ chức hệ luật dẫn với vế phải chỉ là sự kiện:

**p1, p2, …, pm -> q**

Mô hình biểu diễn tri thức dạng này có những ưu điểm là dễ hiểu, dễ giải thích vì các khái niệm được mô tả có cấu trúc tương đối đơn giản. Có thể xây dựng cơ chế suy luận dễ dàng bằng cách áp dụng phương pháp duy diễn tiến hay lùi. Ngoài ra, các luật được biểu diễn theo mô hình này thường độc lập nhau nên việc cập nhật luật, hiệu chỉnh và bảo trì hệ trì hệ thống thuận lợi. Tuy nhiên, từ những đặc điểm trong cách biểu diễn tri thức dạng này tạo nên ưu điểm thì nó cũng tạo ra các khuyết

điểm sau: Chính vì các sự kiện có cấu trúc đơn giản, trong khi tri thức của một số lĩnh vực lại trừu tượng và phức tạp, các khái niệm của lĩnh vực có quan hệ ràng buộc

lẫn nhau nên mô hình biểu diễn này không thể hiện được hết tất cả những yếu tố

trừu tượng đó.

Nói về hệ luật dẫn đối với tri thức tư vấn máy tình bên em có 2 dạng luật dẫn

2.5ĐỘNG CƠ SUY DIỄN

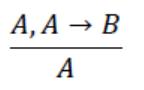
Các cơ sở tri thức đều có động cơ suy diễn để tiến hành các suy diễn để tạo ra các

tri thức mới dựa trên các giả thuyết, sự kiện cung cấp từ ngoài vào và tri thức có sẵn trong hệ cơ sở tri thức.

Động cơ suy diễn thay đổi theo độ phức tạp của cơ sở tri thức. Các hệ cơ sở tri thức làm việc theo cách được điều khiển bởi dữ liệu sẽ dựa vào các thông tin sẵn có( sự kiện cho trước) và tạo ra các sự kiện mới được suy diễn.

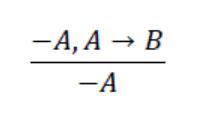
Kỹ thuật được dùng trong hệ thống này là suy diễn tiến.Kỹ thuật này dựa trên suy diễn logic:

+ Modus ponens(MP) : các luật suy diễn đơn giản gọi là "modus ponens".



Nghĩa là A đúng và A  B đúng thì B đúng.

+ Modus Tollens(MT):



Nghĩa là nếu B sai và A  B đúng thì A cũng sai.

Dựa trên các tập kéo theo tức là các luật, và các dữ liệu ban đầu, luật modus ponens tạo nên một dãy các khẳng định. Quá trình suy diễn được tiến hành nhờ một dãy các thông tin đã được khẳng định. Loại suy diễn này là cơ sở của suy diễn dữ liệu hay hệ chuyên gia suy diễn tiến.

\* **Suy diễn tiến**:

Quá trình giải quyết của suy diễn tiến với bước đầu là thu thập thông tin. Thông tin này được suy diễn để đưa ra kết luận. Điều này cũng như người dùng nhập vào những thông tin về máy tính họ cần dùng tư vấn viên ( những chuyên gia ) sẽ tư vấn cho họ mua máy tính nào là thích hợp

Chiến lược suy diễn bắt đầu bằng tập sự kiện đã biết, rút ra các sự kiện mới nhờ dùng các luật mà phần giả thuyết khớp với sự kiện đã biết, và tiếp tục quá trình này cho

đến khi thấy trạng thái đích, hoặc cho đến khi không còn luật nào khớp được các sự kiện

đã biết hay được sự kiện suy diễn. Hệ thống áp dụng suy diễn tiến hoạt động như sau:

* Bước 1:Khởi tạo bộ nhớ, lấy các thông tin về bài toán từ người sử dụng và đặt chúng vào bộ nhớ làm việc.
* Bước 2: Lặp để tìm luật và cho vào bộ nhớ

o Suy diễn quét các luật theo dãy xác định trước.

o Xem phần giả thuyết có trùng khớp với nội dung trong bộ nhớ ?

o Nếu phát hiện một luật như mô tả trên, bổ sung kết luận của luật này vào bộ nhớ.

o Tiếp tục quá trình này, có thể bỏ qua các luật đã cháy.

1. Quá trình tiếp tục cho đến khi không còn khớp được luật nào. Lúc này bộ nhớ có các thông tin cảu người dùng và thông tin do hệ thống suy diễn.

Ví dụ: Gỉa sự người dùng nhập vào 3 sự kiên, họ muốn mua máy tính có Price = 13 triệu, và RAM=4G , weigh= 2.3.

Hệ thống sẽ đưa các sự kiện vào suy diễn :

Price = 13 triệu có 15 máy Sau đó suy diễn tiếp với RAM=4G thì còn lại 7 máy sau đó sự kiện cuối cùng là weigh=2.3 thi sorf lại còn được 3 máy. Sau đó trả kết quả về cho người dùng

Ưu điểm:

1. Làm việc tốt khi bài toán về bản chất là đi thu thập thông tin rồi thấy điều cần suy diễn.
2. Suy diễn tiến cho ra khối lượng lớn các thông tin từ một số thông tin ban đầu. Nó sinh ra nhiều thông tin mới. Suy diễn tiến là tiếp cận lý tưởng đối với loại bài toán cần giải quyết các nhiệm vụ như chẩn đoán bệnh trong y khoa, lập kế hoạch, điều hành điều khiển và diễn dịch.

Khuyết định:

1. Không cảm nhận được rằng chỉ một vài thông tin là quan trọng. Hệ thống hỏi các câu hỏi có thể hỏi mà không biết rằng chỉ một ít câu đã đi đến kết

luận được

1. Hệ thống có thể hỏi cả câu không liên quan. Có thể các câu trả lời cũng quan trọng, nhưng làm người dùng lúng túng khi phải trả lời các câu không dính chủ đề.

[CHƯƠNG 3: THIẾT KẾ CƠ SỞ TRI THỨC](#page28)

3.1. [THU THẬP VÀ PHÂN LOẠI TRI THỨC](#page28)

3.1.1. [Miền tri thức thu thập](#page28)

Tri thức mà nhóm chúng em thu thập là các tri thức về máy tính, nhứng tính năng của máy tính, cấu hình…

3.1.2. [Nguồn thu thập](#page28)

Nguồn tri thức chúng em thu thập là từ trang web <http://www.logicalincrements.com/>

Trang web này chưa rất nhiều thông tin về máy tính, các thành phần của máy tính từ bàn phím, CPU, Màn hình, …. Giới thiệu rất chi tiết nhưng kèm theo đó là những hạn chế không kém, có ưu điểm nhưng cũng ko kém nhược điêm:

+ trang web bằng tiếng anh một phần nào đó sẽ là rào cản ngôn ngữ

+ trang web có rất nhiều thông tin có thể đây là một thuận lợi đối với người hiểu biết máy tính nhiều nhưng phần còn lại không còn là thuận lợi nữa mà sẽ làm cho họ bị rối thông tin, họ không biêt nhiều về nó, họ sẽ cảm thấy chán.

3.1.3. [Cách thức thu thập](#page29)

Đối với bài toán thu thập thông tin nhóm chung em có cách giải quyêt, nhóm em xây dựng 1 bộ crawl dữ liệu về, thông tin lấy về là những thông tin thô chưa được xử lý, bao hồm 2 miền tri thức nhỏ

+ PC

+ Laptop

Cách chúng em lấy dữ liệu về và xử lý dữ liệu nhóm chúng em xin trình bày kĩ ở phần tổ chức tri thức.

3.1.4. [Kết quả thu thập](#page30)

Dữ liệu lấy về là 2 miền tri thức nhỏ thô chưa được xử lý được lưu trong file

+ pc.JSON

+ laptop.JSON

File dữ liệu sau khi lấy về có dạng như thế này:



3.2. [MÔ HÌNH TRI THỨC CỦA ỨNG DỤNG](#page52)

3.2.1. [Tập khái niệm C](#page53)

Mô hình tri thức KOBE thể hiện các tri thức về lính vực máy tính

**KOBE = LAPTOP PC**

**PC=(C1, Rela1, Rule1, attr1,Invidual)**

**Laptop= ( C2,Rela2,Rule2, attr2,Invidual2)**

Trong đó:

KOBE : là tên của mô hình được đặt của miền tri thức máy tính, mô hình được xây dựng của trên mô hình COKB

LAPTOP: là mô hình thể hiện cho tri thức Laptop

PC : là mô hình thể hiện cho tri thức PC

* C:tập hợp các khái niệm về máy tính
* Rela: tập hợp các quan hệ giữa các thành phần khái niệm
* Rule: tập hợp các luật suy diễn trên các sự kiện.

Các thành phần trong mô hình được mô tả chi tiết dưới đây:

3.2.2. [Tập](#page54) các thành phần cho miền tri thức PC

C: là tập các thành phần khái niệm biểu diễn cho miền tri thức PC

C= {CPU, Case, GPU, HDD, HSF, MOBO, PC, PSU, RAM, SSD, TIER, ..} C= | 15|

Attr : Tập thuộc tính thể hiện cho miền tri thức PC

Attr={ Decription, Image, Name, Link, Price, Total,.. } Attr= |8|

Ví dụ

CPU có thể hiện là CPU1 = { Decription=” A dual-core 14nm Skylake, with hyperthreading (handles 4 threads). Great single-threaded performance for a very cheap price. Make sure that the motherboard (particularly the BIOS version) you buy can overclock, if you intend to overclock”, Image =” <http://ecx.imagesamazon.com/images/I/71TAF2Z73uL._SL400_.jpg> ”, Link : ” [http://www.amazon.com/Intel-BX80662I36100-i3-6100-CacheProcessor/dp/B015VPX2EO/” Name](http://www.amazon.com/Intel-BX80662I36100-i3-6100-CacheProcessor/dp/B015VPX2EO/) =” i3 6100”, Price =” 117”,..}

Invidual : Tập các thể hiện của các lớp trong khái niệm , ví dụ quả cam là thể hiện cho lớp hoa quả, ở đây tương tự cũng vậy mỗi class sẽ có những Invidual thể hiển cho class đó.

Invidual={ CPU1, CPU2,CPU2,…Case1,Case2,Case2,…..} Invidual = |200|

CPU1, CPU2, CPU3,… là thể hiện của CPU

Case1, Case2,Case3,…. Là thể hiện của Case

….

Rela : Mỗi quan hệ giữa Class với Class, Class với Invidual

Rela = { hasCPU, hasCase, hasGPU, hasHDD, hasHSF, hasMOBO, hasPC, hasPSU, hasRAM, hasSSD, Hastier, HasDecription, HasImage, HasName, HasLink, HasPrice, HasTotal ,…} Rela = |35|

Ví dụ : CPU hasCPU CPU1 , CPU hasCPU CPU2, CPU hasCPU CPU3, CPU hasCPU CPU4,….

Case hasCase Case1, Case hasCase Case2, Case hasCase Case3,…

…

1. Tập các thành phần cho miền tri thức Laptop

C: là tập các thành phần khái niệm biểu diễn cho miền tri thức PC

C= {CPU,GPU.} C= | 15|

Attr : Tập thuộc tính thể hiện cho miền tri thức PC

Attr={ Decription, Image, Name, Link, Price, Total,.. } Attr= |8|

Ví dụ

CPU có thể hiện là CPU1 = { Decription=” A dual-core 14nm Skylake, with hyperthreading (handles 4 threads). Great single-threaded performance for a very cheap price. Make sure that the motherboard (particularly the BIOS version) you buy can overclock, if you intend to overclock”, Image =” <http://ecx.imagesamazon.com/images/I/71TAF2Z73uL._SL400_.jpg> ”, Link : ” [http://www.amazon.com/Intel-BX80662I36100-i3-6100-CacheProcessor/dp/B015VPX2EO/” Name](http://www.amazon.com/Intel-BX80662I36100-i3-6100-CacheProcessor/dp/B015VPX2EO/) =” i3 6100”, Price =” 117”,..}

Invidual : Tập các thể hiện của các lớp trong khái niệm , ví dụ quả cam là thể hiện cho lớp hoa quả, ở đây tương tự cũng vậy mỗi class sẽ có những Invidual thể hiển cho class đó.

Invidual={ CPU1, CPU2,CPU2,…Case1,Case2,Case2,…..} Invidual = |200|

CPU1, CPU2, CPU3,… là thể hiện của CPU

Case1, Case2,Case3,…. Là thể hiện của Case

….

Rela : Mỗi quan hệ giữa Class với Class, Class với Invidual

Rela = { hasCPU, hasCase, hasGPU, hasHDD, hasHSF, hasMOBO, hasPC, hasPSU, hasRAM, hasSSD, Hastier, HasDecription, HasImage, HasName, HasLink, HasPrice, HasTotal ,…} Rela = |35|

Ví dụ : CPU hasCPU CPU1 , CPU hasCPU CPU2, CPU hasCPU CPU3, CPU hasCPU CPU4,….

Case hasCase Case1, Case hasCase Case2, Case hasCase Case3,…

…

3.3 TỔ CHỨC TRI THỨC TRÊN MÁY TÍNH

Hai miền tri thức sử dụng chung một chuẩn lưu trữ JSON-LD như sau:

|  |
| --- |
| [  // Tên IRI Ontology  {  "@id":"<Tên-Ontology>",  "@type":[  "http://www.w3.org/2002/07/owl#Ontology"  ]  },  // Danh sách định nghĩa IRI của các lớp  {  "@id":"<Tên-Ontology>#<Tên-Lớp>",  "@type":[  "http://www.w3.org/2002/07/owl#Class"  ]  },  …  // Danh sách định nghĩa IRI của các quan hệ  {  "@id":"<Tên-Ontology>#<Tên-Quan-Hệ>",  "@type":[  "http://www.w3.org/2002/07/owl#ObjectProperty"  ],  // Danh sách các domains  "http://www.w3.org/2000/01/rdf-schema#domain":[  {  "@id":"<Tên-Ontology>#<Tên-Lớp>"  },  …  ],  // Danh sách các ranges  "http://www.w3.org/2000/01/rdf-schema#range":[  {  "@id":"<Tên-Ontology>#<Tên-Lớp>"  },  …  ]  },  …  // Danh sách định nghĩa IRI của các thuộc tính  {  "@id":"<Tên-Ontology>#<Tên-Thuộc-Tính>",  "@type":[  "http://www.w3.org/2002/07/owl#DatatypeProperty"  ],  // Danh sách các domains  "http://www.w3.org/2000/01/rdf-schema#domain":[  {  "@id":"http://uit.edu.vn/ontologies/pc#<Tên-Lớp>"  },  …  ],  // Danh sách các ranges  "http://www.w3.org/2000/01/rdf-schema#range":[  {  "@id":"http://www.w3.org/2001/XMLSchema#decimal"  },  …  ]  },  …  // Danh sách định nghĩa IRI của các kiểu dữ liệu  {  "@id":"http://www.w3.org/2001/XMLSchema#<Kiểu-Dữ-Liệu-XML>",  "@type":[  "http://www.w3.org/2000/01/rdf-schema#Datatype"  ]  },  …  // Danh sách định nghĩa IRI của các thể hiện và các giá trị thuộc tính, quan hệ của thể hiện  {  "@id": “<Tên-Ontology>#<Tên-Thể-Hiện>",  "@type":[  "http://www.w3.org/2002/07/owl#NamedIndividual",  "<Tên-Ontology>#<Tên-Lớp>"  ],  // Danh sách các giá trị thuộc tính  "<Tên-Ontology>#<Tên-Thuộc-Tính>":[  {  // Mặc định nếu tồn tại @type thì @value có kiểu string  "@type":"http://www.w3.org/2001/XMLSchema#<Kiểu-Dữ-Liệu-XML>",  "@value":"<Giá-Trị-Thuộc-Tính>"  }  ],  …  // Danh sách các giá trị quan hệ  "<Tên-Ontology>#<Tên-Quan-Hệ>":[  {  "@id":"<Tên-Ontology>#<Tên-Thể-Hiện>"  }  ],  …  },  …  ] |

* Ví dụ, một ontology về PC được lưu trữ trong tập tin pc.jsonld như sau:

|  |
| --- |
| [  // Tên IRI Ontology  {  "@id":"http://uit.edu.vn/ontologies/pc",  "@type":[  "http://www.w3.org/2002/07/owl#Ontology"  ]  },  // Danh sách định nghĩa IRI của các lớp  {  "@id":"http://uit.edu.vn/ontologies/pc#Case",  "@type":[  "http://www.w3.org/2002/07/owl#Class"  ]  },  {  "@id":"http://uit.edu.vn/ontologies/pc#Pc",  "@type":[  "http://www.w3.org/2002/07/owl#Class"  ]  },  // Danh sách định nghĩa IRI của các quan hệ  {  "@id":"http://uit.edu.vn/ontologies/pc#hasCase",  "@type":[  "http://www.w3.org/2002/07/owl#ObjectProperty"  ],  // Danh sách các domains  "http://www.w3.org/2000/01/rdf-schema#domain":[  {  "@id":"http://uit.edu.vn/ontologies/pc#Pc"  }  ],  // Danh sách các ranges  "http://www.w3.org/2000/01/rdf-schema#range":[  {  "@id":"http://uit.edu.vn/ontologies/pc#Case"  }  ]  },  // Danh sách định nghĩa IRI của các thuộc tính  {  "@id":"http://uit.edu.vn/ontologies/pc#hasName",  "@type":[  "http://www.w3.org/2002/07/owl#DatatypeProperty"  ],  // Danh sách các domains  "http://www.w3.org/2000/01/rdf-schema#domain":[  {  "@id":"http://uit.edu.vn/ontologies/pc#Case"  }  ],  // Danh sách các ranges  "http://www.w3.org/2000/01/rdf-schema#range":[  {  "@id":"http://www.w3.org/2001/XMLSchema#string"  }  ]  },  // Danh sách định nghĩa IRI của các kiểu dữ liệu  {  "@id":"http://www.w3.org/2001/XMLSchema#string",  "@type":[  "http://www.w3.org/2000/01/rdf-schema#Datatype"  ]  },  // Danh sách định nghĩa IRI của các thể hiện và các giá trị thuộc tính, quan hệ của thể hiện  {  "@id": “http://uit.edu.vn/ontologies/pc#Case1",  "@type":[  "http://www.w3.org/2002/07/owl#NamedIndividual",  "http://uit.edu.vn/ontologies/pc#Case"  ],  // Danh sách các giá trị thuộc tính  "http://uit.edu.vn/ontologies/pc#<Tên-Thuộc-Tính>":[  {  // Mặc định nếu tồn tại @type thì @value có kiểu string  "@value":"NZXT Source 210"  }  ]  // Danh sách các giá trị quan hệ  // Case1 không tồn tại quan hệ  },  {  "@id": “http://uit.edu.vn/ontologies/pc#Pc1",  "@type":[  "http://www.w3.org/2002/07/owl#NamedIndividual",  "http://uit.edu.vn/ontologies/pc#Pc"  ],  // Danh sách các giá trị thuộc tính  // Pc1 không tồn tại thuộc tính  // Danh sách các giá trị quan hệ  "http://uit.edu.vn/ontologies/pc#hasCase":[  {  "@id":"http://uit.edu.vn/ontologies/pc#Case1"  }  ]  }  ] |

CHƯƠNG 4: XÂY DỰNG BỘ SUY DIỄN

4.1 CÁC BÀI TOÁN ĐẶT RA CẦN GIẢI QUYẾT

Có 2 dạng bài toán đặt ra :

+ bài toán 1

+ bài toán 2 :

4.2 PHÂN LOẠI VÀ SO KHỚP SỰ KIỆN

Hiện tại vì không có thời gian, dữ liệu con thiếu thốn nên nhóm em xây dựng miền tri thức này có 1 dạng sự kiên :

Title : Value

Qúa trình chuyển hóa dữ liệu:

Qúa trình so khớp sự kiện :

Sự kiện nhập vào được lưu vào tập Known := các sựu kiện Inputs nhập vào sau đó duyệt từng máy tính sau đó só khớp theo dạng :

Title = title

Value = value;

Duyệt tất cả máy tính, tìm những chiêc máy tính có sự kiện nằm trong Known.

4.3 CHIẾN LƯỢC SUY DIỄN

CHƯƠNG 5: XÂY DỰNG ỨNG DỤNG TƯ VẤN MUA MÁY TÍNH

5.1 MỤC TIÊU ỨNG DỤNG

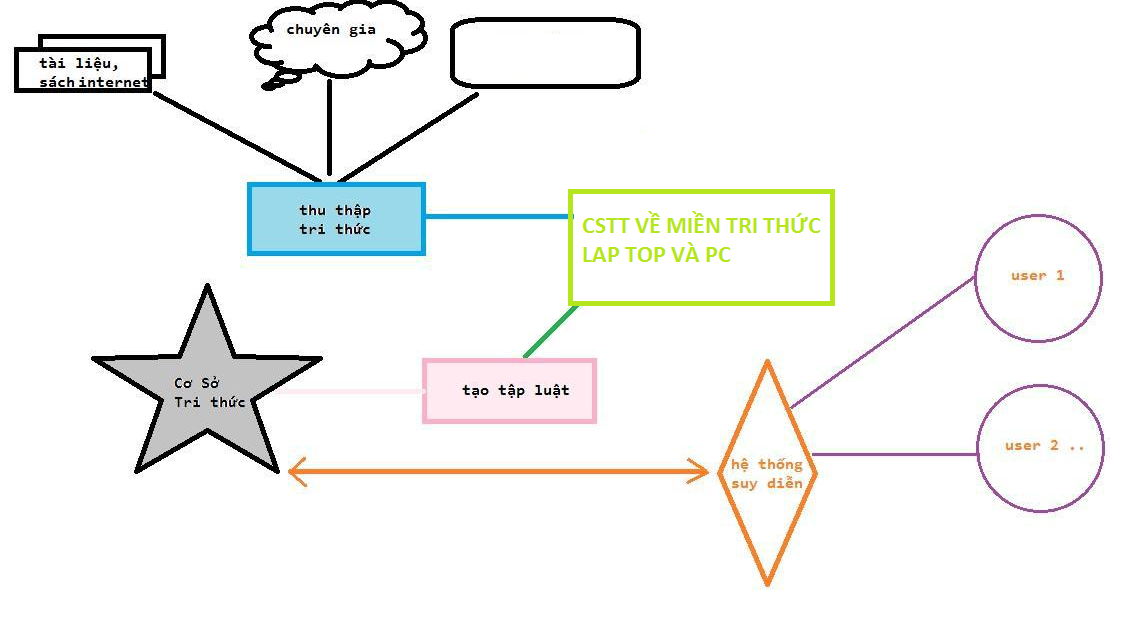
Mục tiêu của ứng dụng là xây dựng một chương trình có khả năng ứng dựng vào thực tế thay cho con người, khi người dùng mua máy tính hiện tại họ nắm số tiền, cấu hình máy, mô trường làm việc.. đó là những sự kiện họ đang nhận nhưng họ muốn xe một máy tính có hợp ý mình không thay vì phải đi đến cửa hàng họ chỉ việc ở nhà trên chiếc máy tính của họ làm việc đó một cách dễ dàng và nhanh chóng.

5.2 YÊU CẦU VÀ CHỨC NĂNG CỦA HỆ THỐNG

Yêu cầu của hệ thống là có khả năng tư vấn chính xác cao.

Chức năng của hệ thống là tư vấn trên 2 miền tri thức nhỏ là PC, client.

5.3 KIẾN TRÚC CỦA HỆ THỐNG



5.4 NỀN TẢNG VÀ CÔNG NGHỆ SỬ DỤNG

Hệ thống được xây dựng trên nền tảng .NET Framework 4.5.1, công nghệ hỗ trợ xây dựng giao diện người dùng WPF.

Công cụ sử dụng:



Visual Studio 2015

 Protégé 5.0.1

.5 TỔ CHỨC GIAO DIỆN

5.6 CÁC HÀM XỬ LÝ

CHƯƠNG 6: THỬ NGHIỆM

6.1 Cách tiến hành thử ngiệm

6.2 kết quả thử nghiệm

6.2 Nhận xét và đánh giá

CHƯƠNG 7: KẾT LUẬ N

7.1 Kết luận của đề tài

Nhìn chung, qua kết quả của các thử nghiệm hệ thống và cũng như là trong quá trình

thực hiện, hệ thống về cơ bản đã đạt được một số yêu cầu như sau:

* + Nghiên cứu về việc thu thập, tổ chức tri thức
  + Mô hình tri thức
  + Sử dụng protégé để biểu diễ tri thức
  + Thực hiện suydiễn
  + Thực hiện được chức năng tra cứu và tư vấn mua máy tính

7.2 Hạn chế của đề tài

Có lẽ vì thời gian làm quá ngắn, có nhiều chuyện phải làm nên có rất nhiều hạn chế như sau :

+ Ngôn ngữ của để tài là tiếng anh nên đã mất đi tính phổ biến

+ giao diện chưa đẹp mắt

+ các luật sự kiện, tri thức còn hạn chế, chính vì điều đó làm giảm tính suy diễn trong hệ thống =>suy diễn chưa đủ mạnh

7.3 Hướng phát triển của đề tài

Hướng phát triển của nhóm đối với ứng dụng này là xây dựng một hệ thống có

+ giao diện thân thiện với người dùng

+ ứng dụng được sử dụng trên nhiều nền tảng

+ giải quyết được vấn đề tính phổ biến và suy diễn chắc chắn

Về lâu nhóm còn có ý định phát triển đề tài thành dự đoán hỏng hóc máy tính, vì đây là cái đầu tiên nhóm hướng đến nhưng vì thời gian không cho phép dữ liệu không đủ nên nhóm đã chuyển sang tư vấn mua máy tính