

MỤC LỤC

PHẦN A: CƠ SỞ TRI THỨC (KNOWLEDGE BASE)	1
I- GIỚI THIỆU TỔNG QUAN VỀ CƠ SỞ TRI THỨC.....	1
I.1-Định nghĩa về cơ sở tri thức:	1
I.2-Các khái niệm.....	1
I.2.1 Thông tin:.....	1
I.2.1 Tri thức	1
I.3- Các thành phần của một hệ ra hỗ trợ quyết định	2
II-LOGIC MỆNH ĐỀ	3
II.1-Giới thiệu về LOGIC.....	3
II.2-Cú pháp LOGIC mệnh đề	3
II.3-Độ ưu tiên.....	3
II.4- Ngữ nghĩa:.....	4
II.5-Các luật ngữ nghĩa	4
II.6-Suy dẫn.....	5
III-HỢP GIẢI MỆNH ĐỀ	8
III.1-Hội chuẩn CNF	8
III.2- Thuật toán hợp giải Robinson.....	9
III.3- Suy diễn tiến và suy diễn lùi	9
III.3.1-Suy diễn tiến	9
III.3.2-Suy diễn lùi	12
IV-LOGIC BẬC NHẤT.....	14
IV.1-LƯỢNG TỪ VỚI MỌI.....	15
IV.2- LƯỢNG TỪ TỒN TẠI.....	15
IV.3- PHÉP THỂ.....	16
IV.4- PHÉP ĐỒNG NHẤT	16
IV.5- ĐỒNG NHẤT TỔNG QUÁT	17
IV.6- ĐỒNG NHẤT TỔNG QUÁT NHẤT	17
V-SUY DIỄN VỚI LOGIC BẬC NHẤT	18
V.1- Cơ sở của hợp giải FOL	18
V.2- Suy diễn tiến và suy diễn lùi	18
V.2.1-Thuật toán suy diễn tiến	19
V.2.2-Thuật toán suy diễn lùi.....	19
V.2.3-Đặc điểm của suy diễn lùi.....	19
PHẦN B : ỨNG DỤNG GRAPHBRAIN.....	21
I-Tổng quan:.....	21
II-Cài đặt :	21

IV- Siêu dữ liệu ngữ nghĩa (The Semantic Hypergraph)23

V- Knowledge Agents.....25

VI- MINH HỌA CÁCH THỰC HIỆN CÁC TÁC VỤ CỦA GRAPHBRAIN.26

1- Phân tích một câu:26

2- Cách thức hoạt động của hyperedge26

3- Phân tích cú pháp và thêm hyperedge vào siêu dữ liệu27

VIII-TÀI LIỆU THAM KHẢO28

PHẦN A: CƠ SỞ TRI THỨC (KNOWLEDGE BASE)

I- GIỚI THIỆU TỔNG QUAN VỀ CƠ SỞ TRI THỨC

I.1-Định nghĩa về cơ sở tri thức:

- Hệ hỗ trợ ra quyết định được Michael S. Scô Morton đề xuất vào những năm 1970. Hệ gồm một số phần chính như: phần mềm máy tính; chức năng hỗ trợ ra quyết định; dữ liệu giao dịch, các mô hình. Có những đặc thù riêng giữa hệ hỗ trợ ra quyết định và hệ thống thông tin. Cụ thể như sau:.

- Hệ thống thông tin có các tính chất.

- Tập trung vào thông tin, hướng đến các nhà quản lý cấp điều hành.

- Làm việc với dòng thông tin có cấu trúc

Các hệ hỗ trợ quyết định có các tính chất:

- Hướng đến các quyết định, các nhà lãnh đạo

- Tính uyển chuyển, thích ứng với hoàn cảnh và phản ứng nhanh

- Do người dùng khởi động và kiểm soát

- Hỗ trợ các quyết định cá nhân của lãnh đạo

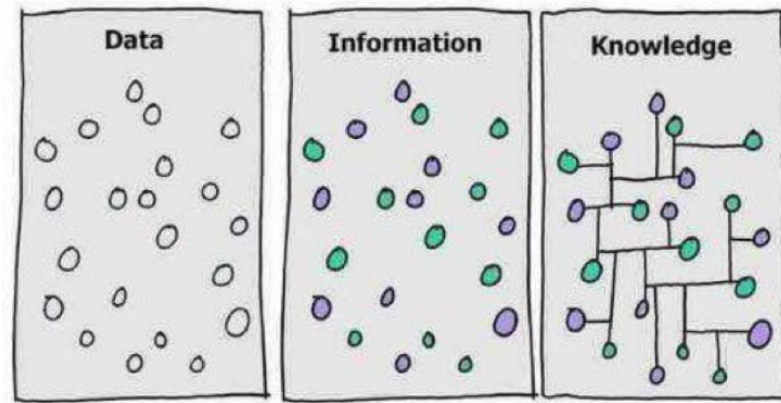
I.2-Các khái niệm

I.2.1 Thông tin:

- **Thông tin:** Những sự kiện được biểu diễn có hệ thống trong một ngữ cảnh cho trước.

I.2.1 Tri thức

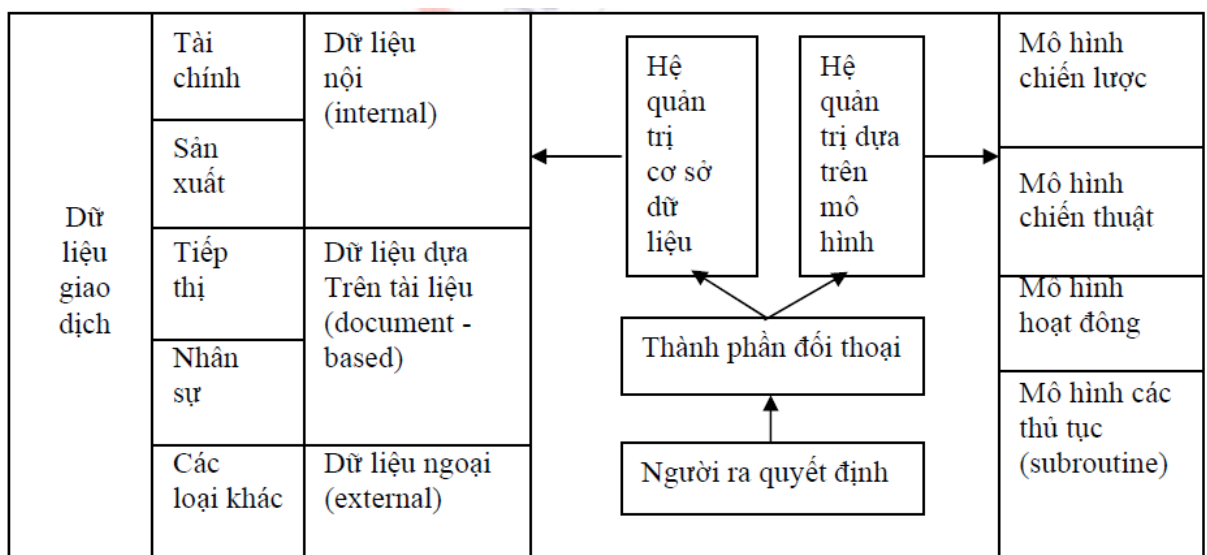
- **Tri thức:** Thông tin phù hợp và có mục tiêu được rút trích từ kinh nghiệm.



Hình 1: Biểu diễn thông tin và tri thức

I.3- Các thành phần của một hệ ra hỗ trợ quyết định

Một cách hình dung về các thành phần của một hệ hỗ trợ ra quyết định (DDS - decision support system) (Hình 2) và quan hệ giữa chúng là sử dụng các khái niệm đối thoại (dialog), dữ liệu (data) mà mô hình (model). Đối với những người thiết kế hệ thống DDS cũng như những người sử dụng hệ thống, điều quan trọng là hiểu được các thành phần này được thiết kế như thế nào. Người sử dụng cần phải biết có thể yêu cầu cái gì ở DDS. Người thiết kế phải biết được DDS có thể cung cấp cái gì.



Hình 2: Các thành phần của một hệ ra quyết định

II-LOGIC MỆNH ĐỀ

II.1-Giới thiệu về LOGIC

- Cần một công cụ để biểu diễn và sử dụng tri thức của con người

Mệnh đề “Nếu trời mưa (A) thì đất ướt (B) “ được mô tả: $A \Rightarrow B$

- Logic: “khoa học về lập luận, chứng minh, suy nghĩ hay suy diễn”
- Sử dụng logic làm một công cụ để biểu diễn và xử lý tri thức

II.2-Cú pháp LOGIC mệnh đề

- Cú pháp: Là những gì được cho phép viết
 - $(C++)$: for (int i=0; i< n; i++)...
 - (Tiếng Việt): Cơm ăn tôi rất ngon.
- Câu (sentence) trong logic mệnh đề:
 - true và false là các câu
 - Các biến mệnh đề là các câu: P, Q, R, Z
 - Nếu α, β là các câu thì
 - $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta$ cũng là các câu
 - Ngoài ra, không có một câu nào nữa

II.3-Độ ưu tiên

$\neg$	Cao nhất	$A \vee B \wedge C$	$A \vee (B \wedge C)$
$\wedge$		$A \wedge B \Rightarrow C \vee D$	$(A \wedge B) \Rightarrow (C \vee D)$
$\vee$		$A \Rightarrow B \vee C \Leftrightarrow D$	$(A \Rightarrow (B \vee C)) \Leftrightarrow D$
$\Rightarrow$	Thấp nhất		
$\Leftrightarrow$			

- Luật ưu tiên cho phép các dạng viết tắt của các câu, nhưng chính thức chỉ có dạng đầy đủ dấu ngoặc mới hợp lệ.

- Các dạng nhập nhằng về cú pháp được cho phép viết tắt chỉ khi chúng tương đương ngữ nghĩa:

$A \wedge B \wedge C$ tương đương với $(A \wedge B) \wedge C$ và $A \wedge (B \wedge C)$

II.4- Ngữ nghĩa:

- Nghĩa của một câu là một chân trị {t, f}
- Thể hiện là việc gán một các chân trị cho các biến mệnh đề
 - $holds(\alpha, i)$ [câu α là **t** trong thể hiện i] [câu α đúng trong thể hiện i]
 - $fails(\alpha, i)$ [câu α là **f** trong thể hiện i] [câu α sai trong thể hiện i]

II.5-Các luật ngữ nghĩa

- $holds(\underline{true}, i)$ với mọi i
- $fails(\underline{false}, i)$ với mọi i
- $holds(\neg\alpha, i)$ nếu và chỉ nếu (*iff*) $fails(\alpha, i)$
- $holds(\alpha \wedge \beta, i)$ *iff* $holds(\alpha, i)$ **và** $holds(\beta, i)$
nối liền
- $holds(\alpha \vee \beta, i)$ *iff* $holds(\alpha, i)$ **hay** $holds(\beta, i)$
nối rời

Thể hiện i dưới dạng bảng tra, P là biến mệnh đề:

- $holds(P, i)$ *iff* $i(P) = t$
- $fails(P, i)$ *iff* $i(P) = f$

Một số dạng viết tắt quan trọng

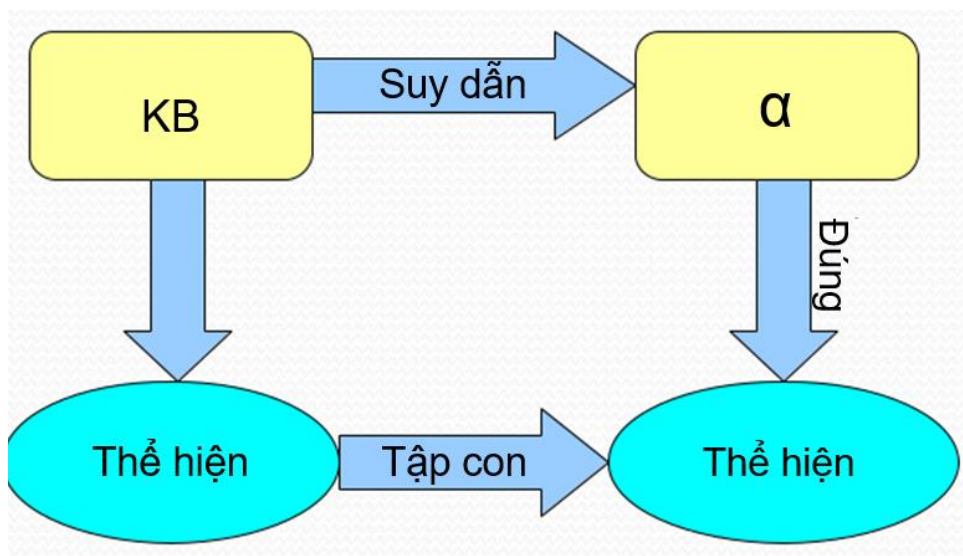
- $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$ (điều kiện, kéo theo) tiền đề $\Rightarrow$ kết luận
- $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ (tương đương)

Bảng chân trị

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$Q \Rightarrow P$	$P \Leftrightarrow Q$
f	f	t	f	f	t	t	t
f	t	t	f	t	t	f	f
t	f	f	f	t	f	t	f
t	t	f	t	t	t	t	t

II.6-Suy dẫn

- Một cơ sở tri thức (KB) **suy dẫn** (entails) một câu α nếu và chỉ nếu mọi thể hiện làm cho KB đúng cũng làm cho α đúng. Ký hiệu: $KB \models \alpha$

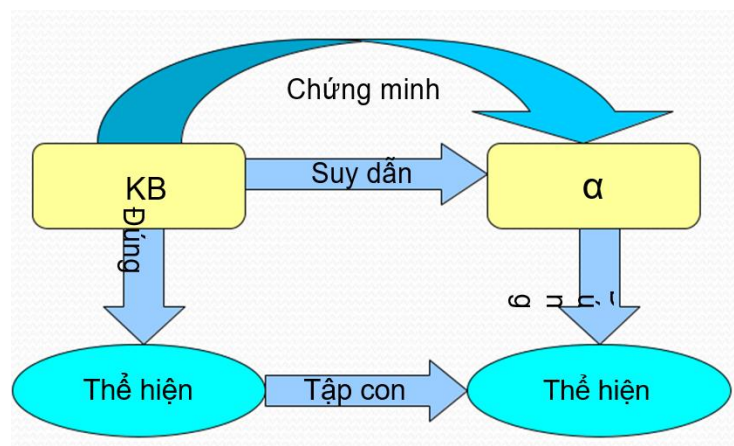


- Tính toán suy dẫn
 - liệt kê tất cả thể hiện
 - chọn những thể hiện mà tất cả thành phần của KB là đúng
 - kiểm tra xem α có đúng trong tất cả các thể hiện này không

Ví dụ :

			KB $\xrightarrow{\text{Suy dẫn}}$ α		
α	β	γ	$\alpha \vee \beta$	$\beta \vee \gamma$	$\alpha \vee \gamma$
False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

- Suy dẫn bằng cách liệt kê
- Suy dẫn và chứng minh
 - Chứng minh là cách kiểm tra xem một KB có suy dẫn một câu α hay không mà không cần liệt kê tất cả các thể hiện có thể



- Chứng minh
 - Một chứng minh là một chuỗi các câu
 - Câu đầu tiên là các tiền đề (KB)
 - Sau đó, ta có thể viết được dòng kế tiếp là kết quả của việc áp dụng một luật suy dẫn lên dòng trước
 - Khi α xuất hiện trên dòng, ta đã chứng minh α từ KB
 - Nếu các luật suy dẫn là **đúng**, thì bất kỳ α có thể chứng minh từ KB cũng suy dẫn được bởi KB

- Nếu các luật suy diễn là **đủ**, thì bất kỳ α nào có thể được suy diễn bởi KB cũng có thể được chứng minh từ KB
- Suy diễn tự nhiên, bao gồm một số luật suy diễn

$\frac{\alpha \Rightarrow \beta}{\alpha}$	$\frac{\alpha \Rightarrow \beta}{\neg \beta}$	$\frac{\alpha}{\beta}$	$\frac{\alpha \wedge \beta}{\alpha}$
β	$\neg \alpha$	$\alpha \wedge \beta$	α
Modus ponens	Modus tolens	And-Introduction	And-Elimination

Ví dụ suy diễn tự nhiên

Chứng minh S

Bước	Công thức	Nguồn gốc
1	$P \wedge Q$	Cho trước
2	$P \Rightarrow R$	Cho trước
3	$Q \wedge R \Rightarrow S$	Cho trước
4	P	1 And-Elim
5	R	4,2 Modus Ponens

- Các hệ thống chứng minh:
 - Có nhiều hệ thống suy diễn tự nhiên; chúng thường là các “chương trình kiểm tra chứng minh”, đúng nhưng không đủ
 - Suy diễn tự nhiên dùng nhiều luật suy diễn gây nên một hệ số phân nhánh lớn trong việc tìm một chứng minh.
 - Thông thường, ta cần dùng “chứng minh theo trường hợp” thậm chí còn phân nhánh nhiều hơn
 - VD: cần chứng minh R từ $(P \vee Q)$, $(P \Rightarrow R)$ và $(Q \Rightarrow R)$.
- Hợp giải mệnh đề :
 - Luật hợp giải:

$$\frac{\alpha \vee \beta \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Luật hợp giải đơn lẻ là một hệ chứng minh đúng và đủ
 - Đòi hỏi tất cả các câu được chuyển sang dạng chuẩn hội
- Các hệ thống Logic
- Hệ thống suy diễn tiến
 - Hệ thống suy diễn lùi
 - Hệ thống dựa trên hợp giải

III-HỢP GIẢI MỆNH ĐỀ

- Hợp giải mệnh đề là luật của suy diễn
- Chỉ sử dụng một mình hợp giải mệnh đề (không cần sử dụng các luật khác) có thể xây dựng một chương trình chứng minh lý thuyết đúng và đủ cho tất cả Logic Mệnh đề
- Chỉ hoạt động với biểu diễn dạng hội chuẩn (Conjunctive Normal Form)

III.1-Hội chuẩn CNF

- Công thức Dạng hội Chuẩn (CNF) có dạng:
 - $(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$
 - $(A \vee B \vee \neg C)$ là một clause
 - A, B, $\neg C$ là các literal, mà mỗi cái là một biến hay phủ định của một biến
 - Mỗi clause phải được thoả và có thể được thoả theo nhiều cách
 - Mỗi câu trong logic mệnh đề đều có thể viết dưới dạng CNF
- Biến đổi thành CNF
 - Loại bỏ các dấu mũi tên ($\Leftarrow, \Leftrightarrow, \Rightarrow$) bằng định nghĩa
 - Đưa dấu phủ định vào dùng luật De Morgan

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$

- Phân phối **or** vào **and**

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

- Mọi câu đều có thể được biến đổi thành CNF, nhưng kích thước có thể tăng lên theo lũy thừa.

III.2- Thuật toán hợp giải Robinson

1. Biến đổi tất cả các câu thành dạng CNF
2. Lấy phủ định kết luận, đưa vào KB
3. Lặp
 1. Nếu trong KB có chứa hai mệnh đề phủ định nhau (p và $\neg p$) thì trả về **true**
 2. Nếu có hai mệnh đề chứa các literal phủ định nhau thì áp dụng hợp giải.
 3. Lặp cho đến khi không thể áp dụng tiếp luật hợp giải.
4. Trả về **false**

III.3- Suy diễn tiến và suy diễn lùi

III.3.1-Suy diễn tiến

- Ý tưởng: kích hoạt tất cả các luật mà tiền đề của nó thỏa trong KB,
 - bổ sung kết luận vào KB, lặp cho đến khi tìm thấy kết luận

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

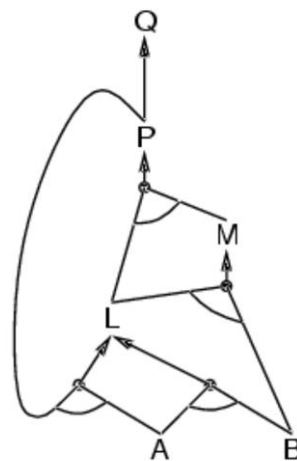
$$B \wedge L \Rightarrow M$$

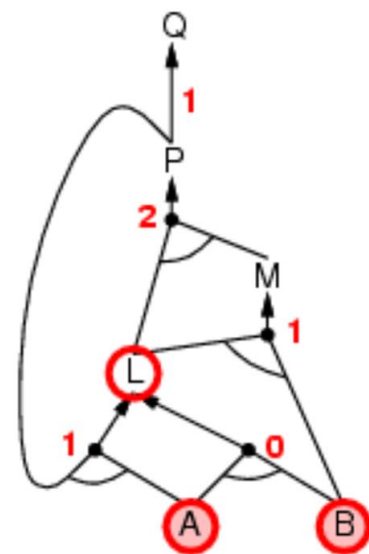
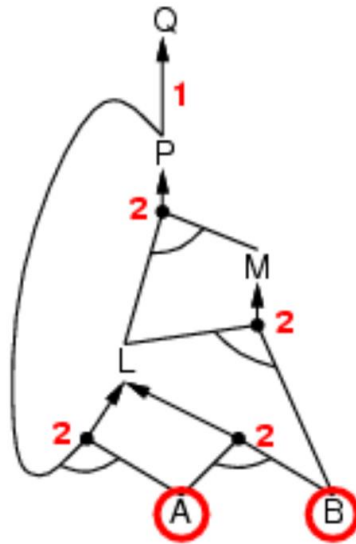
$$A \wedge P \Rightarrow L$$

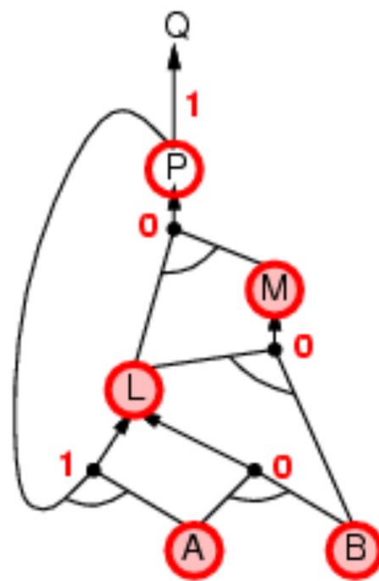
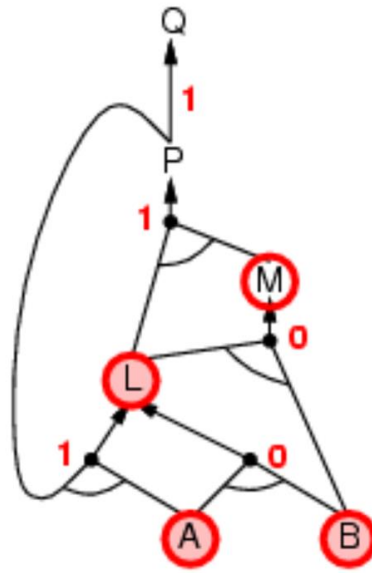
$$A \wedge B \Rightarrow L$$

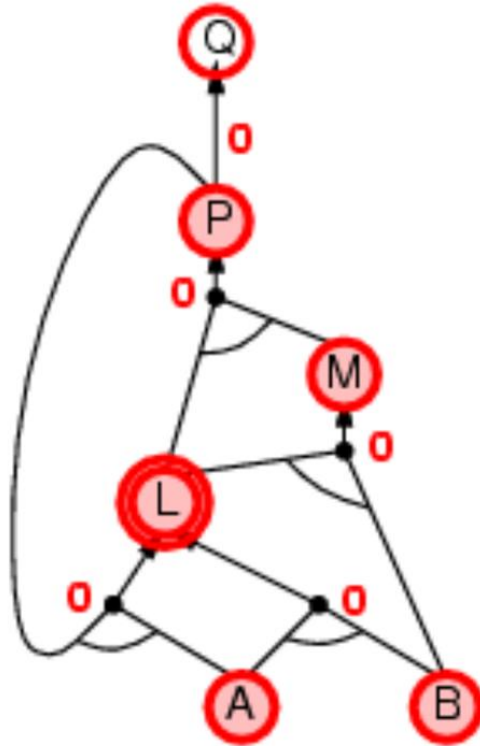
A

B









Kết quả của suy diễn tiến

III.3.2-Suy diễn lùi

Ý tưởng: quay lùi từ câu hỏi q :

để chứng minh q bằng BC,

kiểm tra xem q đã biết chưa, hay

chứng minh bằng cách suy diễn lùi tất cả tiền đề của

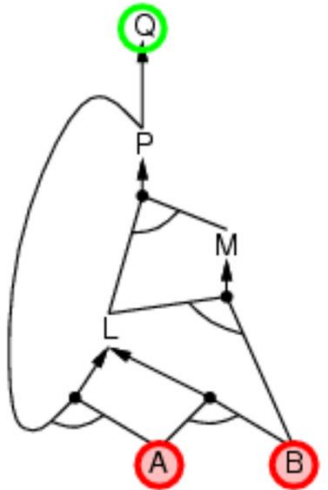
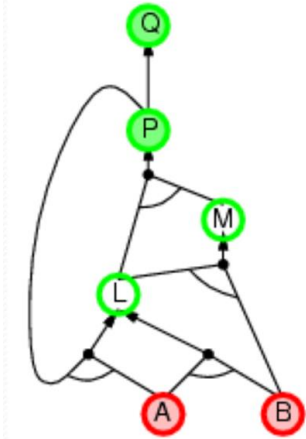
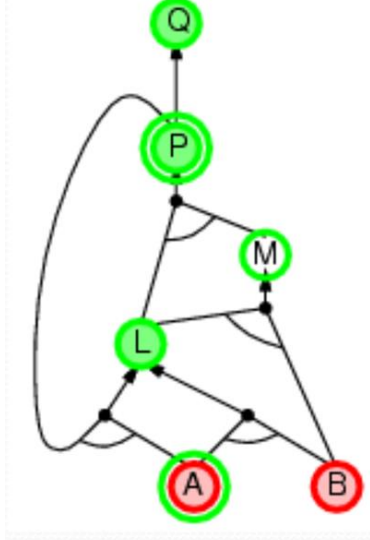
một luật nào đó rút ra q

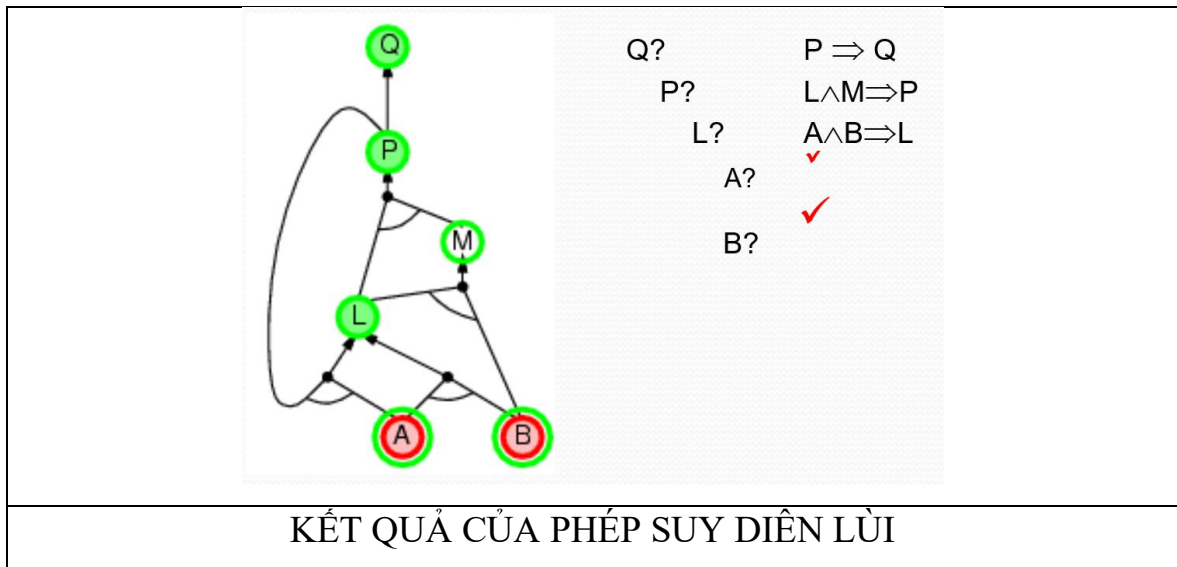
Tránh loop: kiểm tra xem một mục tiêu phụ đã nằm trong ngăn xếp mục tiêu hay chưa

Tránh lặp lại công việc: kiểm tra xem một mục tiêu phụ mới

1. đã được chứng minh đúng, hay
2. đã thất bại

Ví dụ về suy diễn lùi

	Q? P? $P \Rightarrow Q$
Bắt đầu	Lùi dần về
	Q? P? L? $P \Rightarrow Q$ $L \wedge M \Rightarrow P$
Lùi dần về	
	Q? P? L? $P \Rightarrow Q$ $L \wedge M \Rightarrow P$ $A \wedge B \Rightarrow L$ A?
Lùi dần về	



Tóm lại:

- Hiểu được ý tưởng, cơ sở của phép hợp giải và việc chứng minh dùng thuật toán hợp giải
- Nắm được các dạng suy diễn áp dụng được trên logic mệnh đề

IV-LOGIC BẬC NHẤT

- Các câu không thể biểu diễn bằng logic mệnh đề nhưng có thể bằng logic bậc nhất
 - Socrates là người nên socrates chết
 - Khi sơn một hộp bằng màu xanh, nó sẽ trở thành hộp xanh
 - Một người được cho phép truy cập trang web nếu họ được cấp quyền chính thức hay quen biết với ai được phép truy cập
- Biểu thức (Term)
 - Ký hiệu hằng: Lan, Tuan, DHKHTN,...
 - Biến: x, y, a,...
 - Ký hiệu hàm áp dụng cho một hay nhiều term: f(x), tuoi(Lan), anh-cua(Tuan)...

- Câu (Sentence)
 - Một ký hiệu vị từ (predicate) áp dụng cho một hay nhiều *term*: Thuoc(Lan, DHKHTN), La-anh-em(Tuan, Lan), La-ban-be(anh-cua(Tuan), Lan),...
 - $t_1 = t_2$
 - Nếu x là một biến và ϕ là một câu thì $\forall x. \phi$ và $\exists x. \phi$ là một câu
 - Đứng với các toán tử nối câu: $\neg \wedge \vee \leftarrow \leftrightarrow \rightarrow$

IV.1-LƯỢNG TỪ VỚI MỌI

- $\forall <biến> <câu>$

Sinh viên CNTT thì thông minh:

$\forall x \text{ Sinh-viên}(x, \text{CNTT}) \rightarrow \text{Thông-minh}(x)$

- $\forall x P$ đúng trong một mô hình m nếu và chỉ nếu P đúng với x trong mọi đối tượng có thể của mô hình
- Nghĩa là, tương đương với phép **nối liền** của các **thể hiện** của P

$\text{Sinh-viên}(\text{Lan}, \text{CNTT}) \rightarrow \text{Thông-minh}(\text{Lan})$

$\wedge \quad \text{Sinh-viên}(\text{Minh}, \text{CNTT}) \rightarrow \text{Thông-minh}(\text{Minh})$

$\wedge \quad \text{Sinh-viên}(\text{Tuan}, \text{CNTT}) \rightarrow \text{Thông-minh}(\text{Tuan})$

...

- Thông thường, $\rightarrow$ là phép nối thường đi với $\forall$
- Lỗi thường gặp: dùng $\wedge$ làm phép nối chính đi với $\forall$:

$\forall x \text{ Sinh-viên}(x, \text{CNTT}) \wedge \text{Thông-minh}(x)$

nghĩa là “Mọi người đều là sinh viên CNTT và mọi người đều thông minh”

IV.2- LƯỢNG TỪ TỒN TẠI

- $\exists <biến> <câu>$

Có sinh viên CNTT thông minh:

$\exists x \text{ Sinh-viên}(x, \text{CNTT}) \wedge \text{Thông-minh}(x)$

- $\exists x.P$ đúng trong một mô hình m nếu và chỉ nếu P đúng với x trong một đối tượng có thể nào đó của mô hình
- Nghĩa là, tương đương với phép **nối rời** của các **thể hiện** của P

$\text{Sinh-viên}(\text{Lan}, \text{CNTT}) \wedge \text{Thông-minh}(\text{Lan})$

✓ $\text{Sinh-viên}(\text{Minh}, \text{CNTT}) \wedge \text{Thông-minh}(\text{Minh})$

✓ $\text{Sinh-viên}(\text{Tuấn}, \text{CNTT}) \wedge \text{Thông-minh}(\text{Tuấn})$

✓ ...

Lỗi thường gặp

- Thông thường, $\wedge$ là phép nối chính với $\exists$
- Lỗi thường gặp: dùng $\rightarrow$ làm phép nối chính với $\exists$:

$\exists x \text{ Sinh-viên}(x, \text{CNTT}) \rightarrow \text{Thông-minh}(x)$

đúng nếu có bất kỳ ai không là sinh viên

CNTT!

IV.3- PHÉP THẾ

$P(x, f(y), B)$: một câu nguyên tố

Các thể hiện	Phép thế $\{v_1/t_1, v_2/t_2 \dots\}$	Ghi chú
$P(z, f(w), B)$	$\{x/z, y/w\}$	Đổi tên biến
$P(x, f(A), B)$	$\{y/A\}$	
$P(g(z), f(A), B)$	$\{x/g(z), y/A\}$	
$P(C, f(A), B)$	$\{x/C, y/A\}$	Phép thế cơ sở

Áp dụng một phép thế

$P(x, f(y), B) \{y/A\} = P(x, f(A), B)$

$P(x, f(y), B) \{y/A, x/y\} = P(A, f(A), B)$

IV.4- PHÉP ĐỒNG NHẤT

- Hai biểu thức ω_1 và ω_2 là **đồng nhất được (unifiable)** khi và chỉ khi tồn tại thể s sao cho $\omega_1 s = \omega_2 s$
- Gọi $\omega_1 = x$ và $\omega_2 = y$, dưới đây là các **phép đồng nhất**:

S	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	x	x
$\{x/y\}$	y	y
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$	A	A

IV.5- ĐỒNG NHẤT TỔNG QUÁT

- Để đồng nhất $Knows(John, x)$ và $Knows(y, z)$, ta có các phép thể

$$\theta = \{y/John, x/z\} \text{ hay } \theta = \{y/John, x/John, z/John\}$$

- Phép đồng nhất đầu tiên **tổng quát hơn** cái thứ hai.

IV.6- ĐỒNG NHẤT TỔNG QUÁT NHẤT

- g là **phép đồng nhất tổng quát nhất (most general unifier - MGU)** của ω_1 và ω_2 khi và chỉ khi với mọi phép đồng nhất s , tồn tại s' sao cho $\omega_1.s = (\omega_1.g)s'$ và $\omega_2.s = (\omega_2.g)s'$

ω_1	ω_2	MGU
$P(x)$	$P(A)$	$\{x/A\}$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ hay $\{x/y\}$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$
$P(x, f(x))$	$P(x, x)$	Không có MGU!

MỘT SỐ VÍ DỤ VỀ ĐỒNG NHẤT

ω_1	ω_2	MGU
$A(B, C)$	$A(x, y)$	$\{x/B, y/C\}$
$A(x, f(D, x))$	$A(E, f(D, y))$	$\{x/E, y/E\}$
$A(x, y)$	$A(f(C, y), z)$	$\{x/f(C, y), y/z\}$
$P(A, x, f(g(y)))$	$P(y, f(z), f(z))$	$\{y/A, x/f(z), z/g(y)\}$
$P(x, g(f(A)), f(x))$	$P(f(y), z, y)$	Không có
$P(x, f(y))$	$P(z, g(w))$	Không có

V-SUY DIỄN VỚI LOGIC BẬC NHẤT

V.1- Cơ sở của hợp giải FOL

- Hợp giải (Robinson): để chứng minh một tập KB có suy dẫn logic được một câu α hay không, viết lại $KB \wedge \neg \alpha$ dưới dạng mệnh đề (clausal form) và cố gắng suy dẫn ra mệnh đề sai (hợp giải hai mệnh đề đối ngẫu)
- Phép đồng nhất:

$$\text{Unify}(P(x), P(A)) \rightarrow \theta = \{x/A\}$$

Ví dụ :

Chứng minh rằng $(P(x) \Rightarrow Q(x))$ và $P(A)$ suy dẫn logic $\exists z. Q(z)$

1. $\neg P(x) \vee Q(x)$	Tiền đề	
2. $P(A)$	Tiền đề	
3. $\neg Q(z)$	Kết luận	
4. $\neg P(z)$	1, 3	$\theta = \{x/z\}$
5. False	2, 4	$\theta = \{x/z, z/A\}$

V.2- Suy diễn tiến và suy diễn lùi

- Suy diễn tiến (Forward chaining) và suy diễn lùi (Backward chaining) được áp dụng lên các biểu thức dạng Horn
- Biểu thức dạng Horn: trong biểu thức có nhiều nhất một literal khẳng định

$$p_1 \vee \neg p_2 \vee \neg p_3 \vee \dots \vee \neg p_n$$

- Hay dạng luật (luật sinh) $p_2 \wedge p_3 \wedge \dots \wedge p_n \Rightarrow p_1$

V.2.1-Thuật toán suy diễn tiến

```

FOL-FC-Ask(KB,  $\alpha$ ) {
  repeat until new là rỗng
    new  $\leftarrow \{\}$ 
  với mọi câu r trong KB // r ở dạng chuẩn hóa ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ )
    với mọi phép thế  $\theta$  sao cho  $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$ 
      với  $p'_1, \dots, p'_n$  nào đó trong KB
         $q' \leftarrow \text{Subst}(\theta, q)$ 
        if  $q'$  không phải là một câu đã có trong KB hay new
          then thêm  $q'$  vào new
           $\phi \leftarrow \text{Unify}(q', \alpha)$ 
          if  $\phi$  thành công then return  $\phi$ 
  thêm new vào KB
  return false
}

```

V.2.2-Thuật toán suy diễn lùi

```

FOL-BC-ASK(KB, goals,  $\theta$ ) {
  Inputs: KB, cơ sở tri thức
           goals, danh sách dưới dạng nối rời của một câu truy
           vấn  $\theta$ , phép thế hiện tại, được khởi tạo rỗng  $\{\}$ 
  biến cục bộ: ans, một tập các phép thế, được khởi tạo rỗng

  if goals rỗng then return  $\{\theta\}$ 
   $q' \leftarrow \text{SUBST}(\theta, \text{first}(\text{goals}))$ 
  for each r trong KB mà r có dạng chuẩn ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ ) và
     $\theta' \leftarrow \text{UNIFY}(q, q')$  thành công
    ans  $\leftarrow \text{FOL-BC-ASK}(\text{KB}, [p_1, \dots, p_n | \text{REST}(\text{goals})], \theta \cup \theta') \cup \text{ans}$ 
  return ans
}

```

V.2.3-Đặc điểm của suy diễn lùi

- Tìm kiếm chứng minh bằng cách đệ qui theo chiều sâu: không gian tuyến tính theo kích thước của chứng minh
- Không đầy đủ do lặp vô tận
 - Giải pháp: Kiểm tra trạng thái hiện tại với mọi trạng thái đang

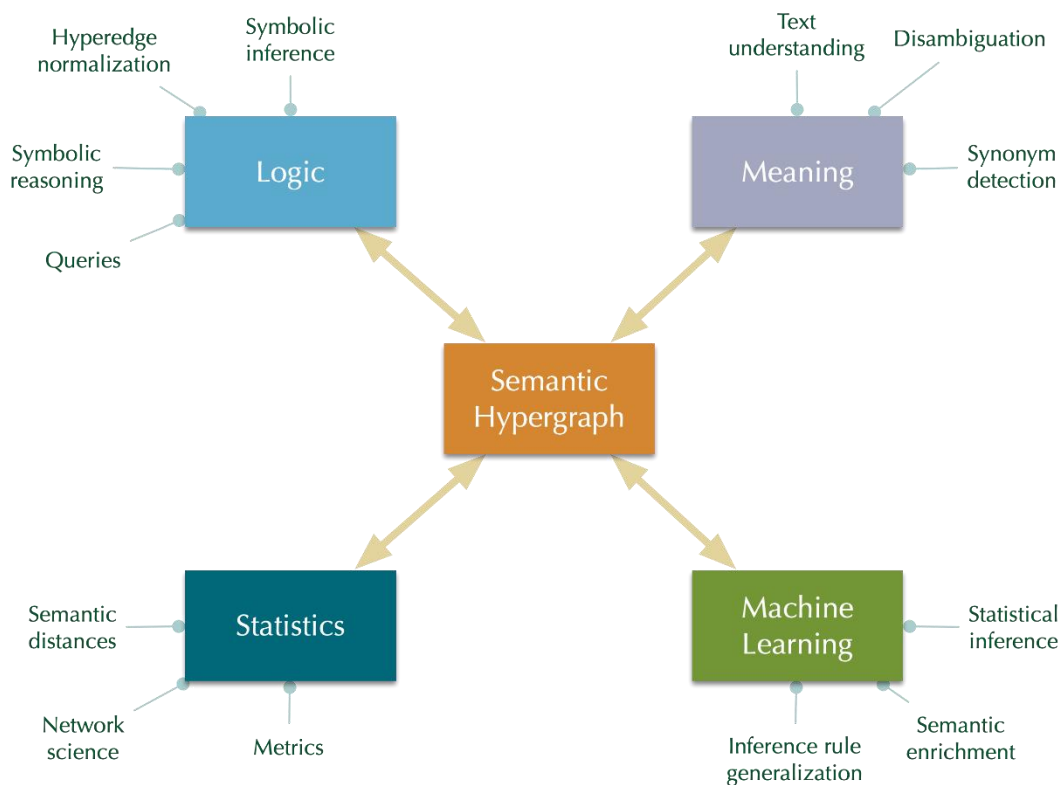
có trong stack

- Không hiệu quả do các mục tiêu con bị lặp lại (cả khi thất bại cũng như thành công)
 - Giải pháp: dùng bộ nhớ tạm lưu lại các mục tiêu con đã duyệt qua
- Được dùng nhiều trong lập trình logic (ngôn ngữ Prolog)

PHẦN B : ỨNG DỤNG GRAPHBRAIN

I-Tổng quan:

- Graphbrain là một thư viện nguồn mở trong lĩnh vực Trí tuệ nhân tạo, vừa là công cụ nghiên cứu khoa học. Mục đích của nó là để tạo điều kiện khai thác ý nghĩa tự động và phân tích văn bản, cũng như khám phá và suy luận về tri thức. Nó là một phần của socsemics, một dự được tài trợ bởi Hội đồng nghiên cứu châu Âu nhằm tập trung vào sự phân mảnh trong không gian cộng đồng trực tuyến.



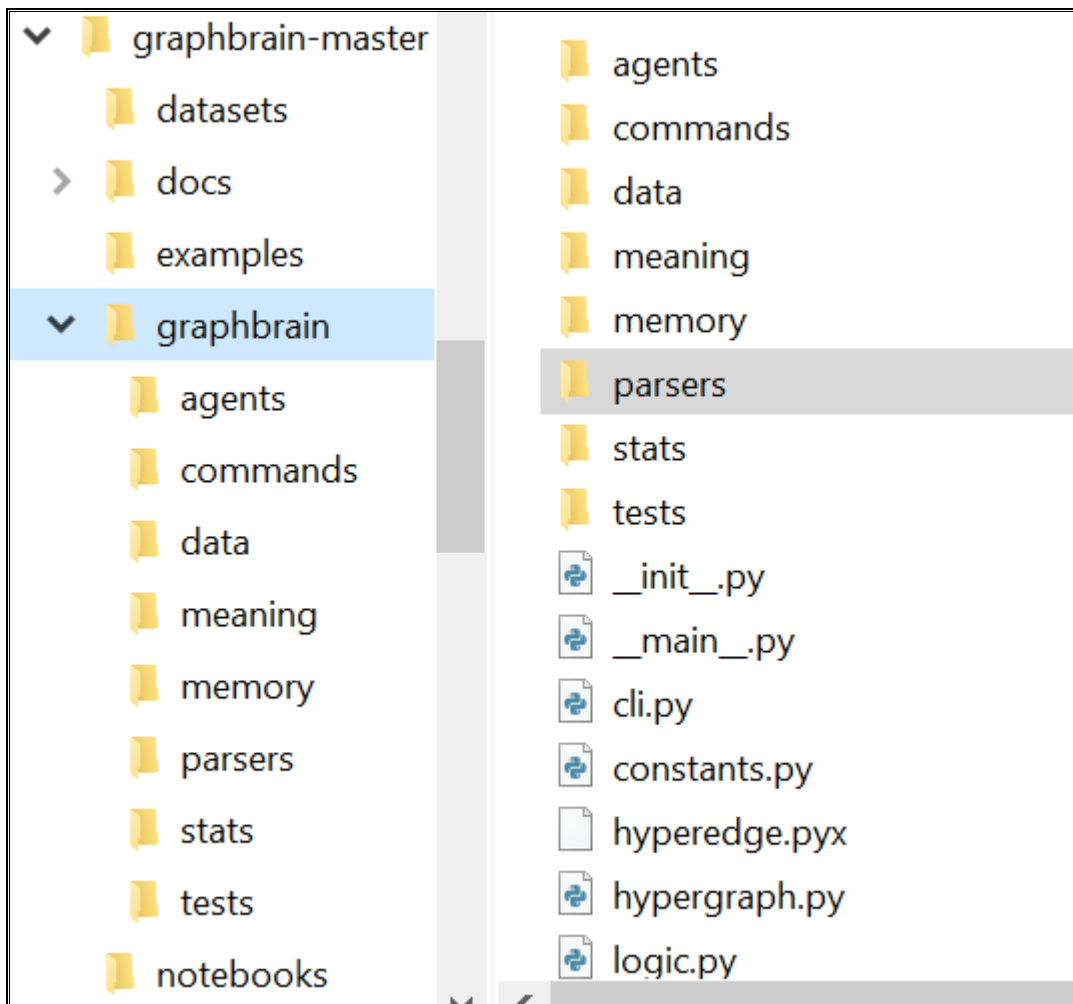
- Graphbrain được viết bằng Python, nhằm tận dụng và tạo điều kiện tích hợp với môi trường phong phú của các thư viện có sẵn trong ngôn ngữ này.

II-Cài đặt :

- \$ pip install graphbrain
- \$ virtualenv -p <path to python3> venv

III-Tổ chức project :

-  datasets
-  docs
-  examples
-  graphbrain
-  notebooks
-  scripts
-  .gitignore
-  AUTHORS
-  CHANGELOG.md
-  LICENSE
-  MANIFEST.in
-  README.md
-  setup.py
-  VERSION



+ Tập Dataset chứa file *.txt

+ Thư mục graphbrain

IV- Siêu dữ liệu ngữ nghĩa (The Semantic Hypergraph)

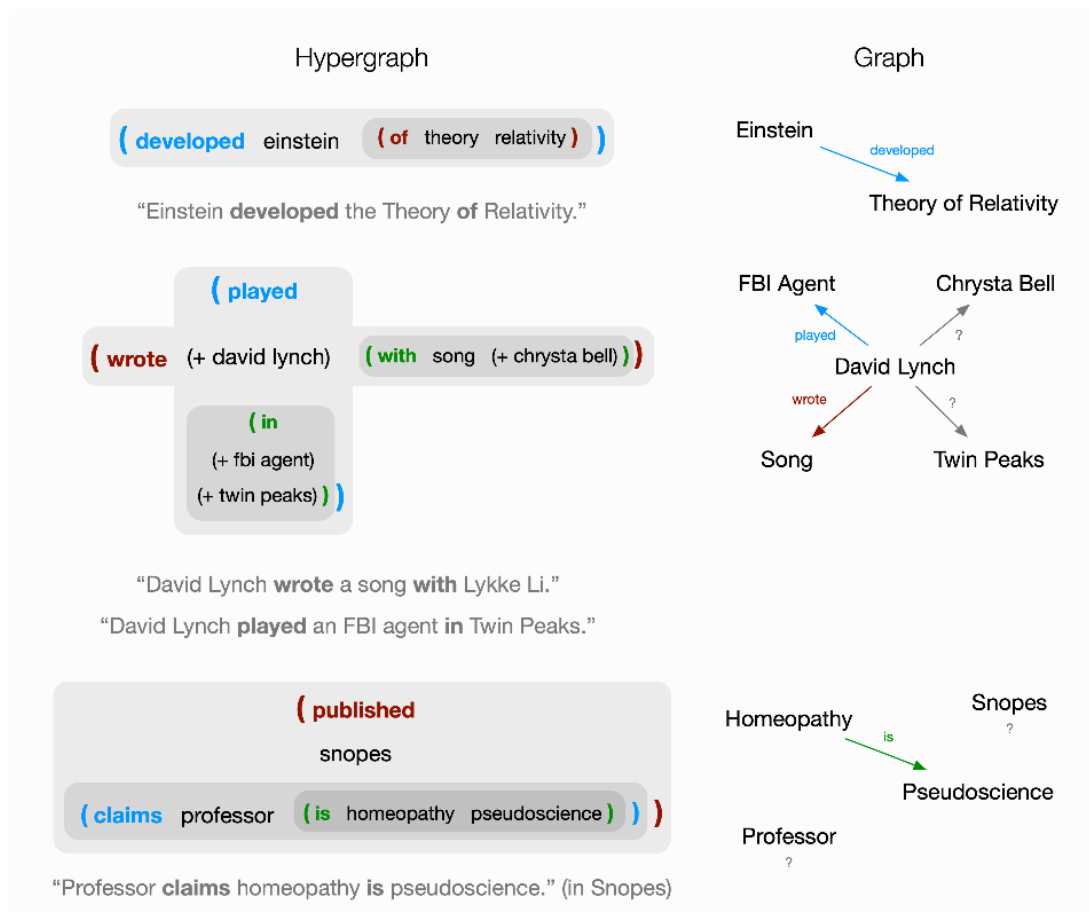
- Siêu dữ liệu ngữ nghĩa là mục đích chính của dự án Graphbrain, cả về khái niệm và chức năng. Có thể nhìn từ ba góc độ khác nhau:
 - * Đóng vai trò như một trung gian giữa ngôn ngữ tự nhiên và hình thức
 - * Như một mô hình kiến thức
 - * Như một cơ sở dữ liệu
- Sự phong phú của thông tin chứa trong ngôn ngữ tự nhiên không thể được nắm bắt và phân tích đầy đủ bằng các phương pháp mạng dựa trên biểu đồ truyền thống hoặc khung phân

phối. Một mặt, ngôn ngữ tự nhiên là đệ quy, cho phép các khái niệm được xây dựng từ các khái niệm khác cũng như các tuyên bố về các tuyên bố và mặt khác, nó có thể diễn đạt các mối quan hệ.

Trong khi một biểu đồ dựa trên một tập hợp các đỉnh và một tập hợp các cạnh mô tả các kết nối dyadic, một siêu đồ thị khái quát cấu trúc như vậy bằng cách cho phép các kết nối -ary.

- **Hyperedge** xác định một cấu trúc ngữ nghĩa bằng cách kết hợp các cấu trúc ngữ nghĩa khác. Mục đích của trình kết nối là xác định theo nghĩa nào các cấu trúc bên trong này được kết nối.
- Ví dụ, như vị ngữ: trong trường hợp này, hyperedge định nghĩa một mệnh đề. Chẳng hạn, câu đơn giản, Berlin Berlin rất đẹp được thể hiện dưới dạng: **(is berlin nice)**
- Để kết hợp các khái niệm theo một cách cụ thể như để xác định một khái niệm mới, ví dụ: thủ đô của Đức: **(of capital germany)**
- Để xây dựng các khái niệm từ các khái niệm khác, ví dụ như thịt và khoai tây: **(and meat potatoes)**
- Hyperedge là một công cụ sửa đổi khái niệm, để thể hiện một ví dụ cụ thể hơn của một khái niệm, ví dụ ngọn núi cao nhất ở Brazil: **(highest (in mountain brazil))**
- Để chỉ định các điều kiện có thể là một phần của một đề xuất, ví dụ: khi bầu trời có màu xanh da trời: **(when (is (the sky) blue))**
- Các cấu trúc này có thể được lồng tùy ý, ví dụ, Mary Mary leo lên ngọn núi cao nhất ở Brazil. **(climbs mary (the (highest (in mountain brazil))))**

+ Có thể minh họa mô hình cơ sở tri thức hyperdge như sau:



Mô hình KB Hypergraph

V- Knowledge Agents

- KA tri thức là các chương trình thu thập siêu dữ liệu theo một cách nào đó. Chúng có thể là tự suy luận (nội suy), chỉ làm việc trên các nội dung hiện tại của siêu dữ liệu để có được kiến thức mới. KA nó có thể suy ra rằng 'mèo đen là một loại' mèo, hay thành phố Berlin, là một 'thành phố'.

Chẳng hạn, KA tạo ra các hyperedges mới như:
(type_of/p/. city/c (of/b city/c berlin/c))

- Một số tác nhân sử dụng các nguồn bên ngoài để giới thiệu kiến thức vào siêu dữ liệu. Ví dụ, tác nhân txt_parser nhận làm đầu

vào một tệp văn bản đơn giản và bộ chuyển đổi mỗi câu mà nó phát hiện trong đó thành Hyperedge.

VI- MINH HỌA CÁCH THỰC HIỆN CÁC TÁC VỤ CỦA GRAPHBRAIN.

Link minh họa sau : <http://graphbrain.net/tutorials.html>

1- Phân tích một câu:

Chuyển đổi một câu trong ngôn ngữ tự nhiên thành một hyperedge là nhiệm vụ cơ bản và tinh túy nhất có thể thực hiện với Graphbrain. Cụ thể trong trường hợp này ngôn ngữ tiếng Anh được lựa chọn.

Code trong python :

```
from graphbrain.parsers import *
parser = create_parser(name='en')
text = "The Turing test, developed by Alan Turing in 1950, is a test of
machine intelligence."
parses = parser.parse(text)
for parse in parses:
    edge = parse['main_edge']
    print(edge.to_str())
```

2-Cách thức hoạt động của hyperedge

Một siêu dữ liệu(hyperedge) cũng có thể được xem như là một loại cơ sở dữ liệu, lưu trữ kiến thức dưới dạng các tập hợp siêu dữ liệu và cung cấp các hàm giúp dễ dàng thêm và tìm kiếm các siêu tăng theo những cách hữu ích. Chúng ta sẽ thấy ở đây làm thế nào để thực hiện một số nhiệm vụ cơ bản với siêu dữ liệu.

```
(is/pd.sc.|f--3s-/en
  [:/b/.
    [the/md/en
      [+/b.mm/. turing/cp.s/en test/cc.s/en]]
    (developed/pc.ax.<pf----/en
      (by/x/en
        [+/b.am/. alan/cp.s/en turing/cp.s/en])
        {in/tt/en 1950/c#/en}})]
  [a/md/en
    [of/br.ma/en
      test/cc.s/en
      [+/b.am/. machine/cc.s/en intelligence/cc.s/en]]])
```

3- Phân tích cú pháp và thêm hyperedge vào siêu dữ liệu

Code python

```
[int]
parser = create_parser(name='en')
text = "Mary is playing a very old violin."

parses = parser.parse(text)
for parse in parses:
    edge = parse['main_edge']
    hg.add(edge)
for edge in hg.all():
    show(edge, style='online')
```

[out]

```
((is/av.|f--3s-/en playing/pd.so.|pg----
/en) mary/cp.s/en [a/md/en [(very/w/en old/ma/en) violin/cc.s/en]])
[(very/w/en old/ma/en) violin/cc.s/en]
[a/md/en [(very/w/en old/ma/en) violin/cc.s/en]]
(is/av.|f--3s-/en playing/pd.so.|pg----/en)
(very/w/en old/ma/en)
a/md/en
is/av.|f--3s-/en
mary/cp.s/en
```

old/ma/en
playing/pd.so.jpg---/en
very/w/en
violin/cc.s/en

VIII-TÀI LIỆU THAM KHẢO

- 1- <https://graphbrain.net/index.html>
- 2- Nguyễn Quốc Huy PhD (PDF files)