TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

-----🙞🙜🕮🙞🙜----- ****

**Suy diễn tiến và lùi vẽ cây luật mô phỏng**

|  |
| --- |
| Giáo viên hướng dẫn: ***PGS. TS. Phạm Văn Hải***  Thực hiện: Ngô Thị Linh - 20187179  Ngô Phú Thái - 20187197  Nguyễn Đức Thâu - 20187199  **Hà Nội, ngày 10/10/2020.** |

Mục lục

[I. Giới thiệu 3](#_Toc53334778)

[II. Kiến thức cơ bản 4](#_Toc53334779)

[1. Logic mệnh đề 4](#_Toc53334780)

[a. Cú pháp: 4](#_Toc53334781)

[b. Ngữ nghĩa 5](#_Toc53334782)

[c. Sự tương đương logic 5](#_Toc53334783)

[2. Chuẩn tắc 6](#_Toc53334784)

[3. Chuẩn CNF, chuẩn DNF 7](#_Toc53334785)

[4. Luật suy diễn hợp giải 8](#_Toc53334786)

[a. Luật suy diễn hợp giải (Resolution) 8](#_Toc53334787)

[b. Giải thuật hợp giải 9](#_Toc53334788)

[c. Chứng minh bằng hợp giải 9](#_Toc53334789)

[5. Suy diễn tiến (Forward chaining) 11](#_Toc53334790)

[6. Suy diễn lùi (Backward chaining) 13](#_Toc53334791)

[7. Suy diễn tiến hay suy diễn lùi? 14](#_Toc53334792)

III. Khảo sát, đặc tả yêu cầu bài toán, giải quyết bài toán……………

1. Áp dụng suy diễn tiến và lùi giải bài toán………..

2. Vẽ cây luật mô phỏng…………….

IV. Kết quả thực nghiệm………………………………..

V. Kết luận………………………………………..……

VI. Tài liệu tham khảo………………………….……….

# Giới thiệu

Loài người thông minh vì biết lập luận. Liệu máy tính có khả năng lập luận được (như con người) không?

Để trả lời câu hỏi này, chúng ta trước hết hãy cho biết thế nào là lập luận. Lập luận là hành động sinh ra một phát biểu đúng mới từ các phát biểu đúng có trước. Hay nói cách khác, một người hoặc một hệ thống được gọi là biết lập luận nếu nó chỉ ra rằng một phát biểu nào đó đúng (true) khi cho trước một tập các phát biểu đúng hay không.

Các phát biểu phải tuân theo một tập các qui tắc nhất định (ngữ pháp) và cách xác định một phát biểu là đúng (true) hay là sai (false). Một tập các qui tắc qui định ngữ pháp và cách xác định ngữ nghĩa đúng/sai của các phát biểu gọi là logic.

Logic là một ngôn ngữ mà mỗi câu trong ngôn ngữ đó có ngữ nghĩa (giá trị) là đúng hoặc sai, và vì vậy có thể cho phép chúng ta lập luận, tức là một câu mới có giá trị đúng khi cho các câu trước đó là đúng hay không. Các câu cho trước được gọi là Cơ sở tri thức (Knowledge base - KB), câu cần chứng minh là đúng khi biết KB đúng gọi là câu Truy vấn (query - q). Nếu q là đúng khi KB là đúng thì ta nói rằng KB suy diễn ra q (ký hiệu là KB ╞ q). Trong bản báo cáo này, chúng ta sẽ xây dựng các thuật giải cho phép lập luận tự động trên các logic khác nhau. Các thuật giải này giúp máy tính có thể lập luận, rút ra phát biểu mới từ các phát biểu cho trước.

# Kiến thức cơ bản

## Logic mệnh đề

### Cú pháp:

* Ký hiệu:
* Hai hằng logic: True và False**.**
* Các ký hiệu mệnh đề (còn được gọi là các biến mệnh đề): P, Q,...
* Các phép kết nối logic: ∧, ∨, ¬, ⇒, ⇔.
* Các dấu mở ngoặc”(“ và đóng ngoặc ”)”.
* **Các quy tắc xây dựng các công thức**

Các biến mệnh đề là công thức. Nếu A và B là công thức thì:

* + (A∧B) (đọc “A hội B” hoặc “A và B”)
  + (A∨B) (đọc “A tuyển B”hoặc “A hoặc B”) (¬A) (đọc “phủ định A”)
  + (A⇒B) (đọc “A kéo theo B”hoặc “nếu A thì B”) (A⇔B) (đọc “A và B kéo theo nhau”) là các công thức.
* Thứ tự ưu tiên của các toán tử logic (từ cao xuống thấp):

¬, ∧, ∨, ⇒, ⇔

### Ngữ nghĩa

True là câu luôn có giá trị đúng, false là câu luôn có giá trị sai.

Mỗi ký hiệu biểu diễn (ánh xạ với) một phát biểu/mệnh đề trong thế giới thực; ký hiệu mệnh đề có giá trị là đúng (true) nếu phát biểu/mệnh đề đó là đúng, có giá trị là sai (false) nếu phát biểu/mệnh đề đó là sai, hoặc có giá trị chưa xác định (true hoặc false).

Ví dụ:

* ¬A có nghĩa là phủ định mệnh đề 🡪 câu A, nhận giá trị true nếu A là false và ngược lại.
* A ∧ B có nghĩa là mối liên kết “A và B”, nhận giá trị true khi cả A và B là true, và nhận giá trị false trong các trường hợp còn lại.
* A ∨ B biểu diễn mối liên kết “A hoặc B”, nhận giá trị true khi hoặc A hoặc B là true, và nhận giá trị false chỉ khi cả A và B là false.
* (A ⇒ B) biểu diễn mối quan hệ “A kéo theo B”, chỉ nhận giá trị false khi A là true và B là false; nhận giá trị true trong các trường hợp khác.
* (A ⇔ B) biểu diễn mối quan hệ “A kéo theo B”, “B kéo theo A”.

### Sự tương đương logic

A ⇒ B ≡ ¬A ∨ B

A ⇔ B ≡ (A⇒B) ∧ (B⇒A)

¬ (¬A) ≡ A

* **Luật De Morgan**

¬ (A ∨ B) ≡ ¬A ∧ ¬B

¬ (A ∧ B) ≡ ¬A ∨ ¬B

* **Luật giao hoán**

A ∨ B ≡ B ∨ A

A ∧ B ≡ B ∧ A

* **Luật kết hợp**

(A ∨ B) ∨ C ≡ A ∨ (B ∨ C)

(A ∧ B) ∧ C ≡ A ∧ (B ∧ C)

* **Luật phân phối**

A ∧ (B ∨ C) ≡ (A ∧ B) ∨ (A ∧ C)

A ∨ (B ∧ C) ≡ (A ∨ B) ∧ (A ∨ C)

## Chuẩn tắc

Câu dạng chuẩn hội là câu hội của các câu tuyển (clause). Như trên đã nói, câu tuyển là câu dạng A­1 ∨ A2 ∨ …∨ An, trong đó các Ai là các ký hiệu mệnh đề hoặc phủ định của ký hiệu mệnh đề. Vậy câu dạng chuẩn hội có dạng:

(A11 ∨ A12 ∨…∨ A1n) ∧ (A21 ∨ A22 ∨…∨ A2m) ∧ … ∧ (Ak1 ∨ Ak2 ∨… ∨Akr)

Với Aij là các literal (là ký hiệu mệnh đề hoặc phủ định của ký hiệu mệnh đề).

Bỏ các dấu kéo theo (⇒) bằng cách thay (A⇒B) bởi (¬A ∨ B).

Chuyển các dấu phủ định (¬) vào sát các ký hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay ¬ (¬A) bởi A.

Áp dụng luật phân phối, thay các công thức có dạng A ∨ (B ∧ C) bởi (A ∨ B) ∧ (A ∨ B).

## Chuẩn CNF, chuẩn DNF

**Dạng chuẩn kết hợp (Conjunctive normal form – CNF)**

* Là kết hợp (liên kết VÀ) của các mệnh đề (clauses).
* Mỗi mệnh đề (clause) là một liên kết HOẶC của các ký hiệu định đề đơn.
* Ví dụ: (p ∨ q) ∧ (¬q ∨ ¬r ∨ s)

**Dạng chuẩn tuyển (Disjunctive normal form – DNF)**

* Là liên kết HOẶC của các mệnh đề (clauses) (clauses).
* Mỗi mệnh đề (clause) là một liên kết VÀ của các ký hiệu định đề

Đơn.

* Ví dụ: (p ∧ ¬q) ∨ (¬p ∧ r) ∨ (r ∧ ¬s)

**Chuyển đổi thành CNF**

* Loại bỏ dấu mũi tên (⇒, ⇔) bằng định nghĩa tương đương.
* Đưa phủ định vào bằng luật De Morgan

¬(A ∨ B) ≡ ¬A ∧ ¬B

¬(A ∧ B) ≡ ¬A ∨ ¬B

* Phân phối or (∨) vào and (∧)

A ∨ (B ∧ C) ≡ (A ∨ B) ∧ (A ∨ C)

* Mọi câu đều có thể chuyển đổi về dạng CNF

**Ví dụ chuyển đổi CNF**

(A ∨ B) ⇒ (C ⇒ D)

1. Loại bỏ mũi tên: ¬ (A ∨ B) ∨ (¬C ∨ D)

2. Đưa phủ định vào: (¬A ∧ ¬B) ∨ (¬C ∨ D)

3. Phân phối: (¬A ∨ ¬C ∨ D) ∧ (¬B ∨ ¬C ∨ D)

## Luật suy diễn hợp giải

### Luật suy diễn hợp giải (Resolution)

* Luật suy diễn hợp giải áp dụng được đối với các biểu thức logic ở dạng chuẩn CNF.
* Luật suy diễn hợp giải có tính đúng đắn (sound), nhưng không có tính hoàn chỉnh (incomplete).
  + Tập giả thiết (cơ sở tri thức) KB chứa biểu thức (p ∧ q)
  + Cần chứng minh: (p ∨ q) ?
  + Luật suy diễn hợp giải không thể suy ra được biểu thức cần chứng minh !
* Nếu các biểu thức trong tập KB và biểu thức (cần chứng minh) α đều ở dạng CNF, thì áp dụng luật suy diễn hợp giải sẽ xác định tính (không) thỏa mãn được của (KB ∧ ¬α).

### Giải thuật hợp giải

* Chuyển đổi tất cả các biểu thức trong KB về dạng chuẩn CNF.
* Áp dụng liên tiếp luật suy diễn hợp giải (Resolution rule) bắt đầu từ: (KB ∧ ¬α)
  + KB là kết hợp của các biểu thức ở dạng chuẩn CNF.
  + Do đó, (KB ∧ ¬α) cũng là một biểu thức ở dạng chuẩn CNF!
* Quá trình áp dụng luật suy diễn hợp giải dừng lại khi:
  + Có mâu thuẫn xảy ra: Sau khi hợp giải, thu được (suy ra) biểu thức rỗng (mâu thuẫn).
  + Không có biểu thức mới nào được sinh ra nữa.

### Chứng minh bằng hợp giải

* Giả sử có tập giả thiết KB.
  + p ∧ q
  + p ⇒ r
  + (q ∧ r) ⇒ s
* Cần chứng minh định lý s.
* Bước 1. Chuyển đổi KB về dạng chuẩn CNF:

(p ⇒ r) được chuyển thành (¬p ∨ r)

((q ∧ r) ⇒ s) được chuyển thành (¬q ∨ ¬r ∨ s)

* Bước 2. Phủ định biểu thức cần chứng minh:

¬s

* Bước 3. Áp dụng liên tiếp luật hợp giải đối với (KB ∧ ¬α):

{p, q, ¬p ∨ r, ¬q ∨ ¬r ∨ s, ¬s}

* Bắt đầu quá trình hợp giải, ta có tập các mệnh đề:

1. p

2. q

3. ¬p ∨ r

4. ¬q ∨ ¬r ∨ s

5. ¬s

* Hợp giải 1. và 3., ta thu được:

6. r

* Hợp giải 2. và 4., ta thu được:

7. ¬r ∨ s

* Hợp giải 6. và 7., ta thu được:

8. s

* Hợp giải 8. và 5., ta thu được mâu thuẫn ( {} ).
* Tức là biểu thức ban đầu (s) được chứng minh là đúng.

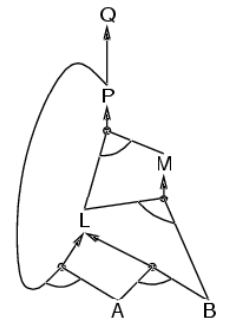
|  |
| --- |
| p q ¬p ∨ r ¬q ∨ ¬r ∨ s ¬s    r  ¬r ∨ s  s    {} (mâu thuẫn) |

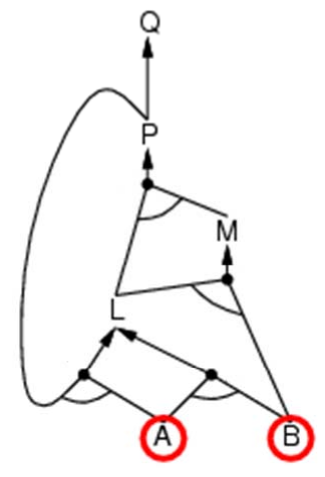
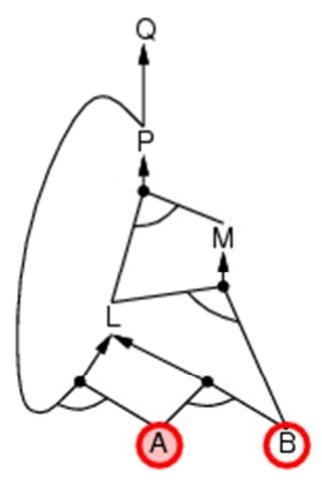
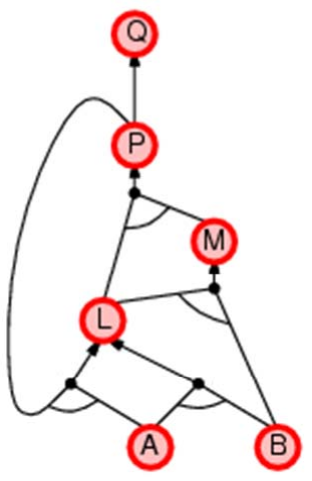
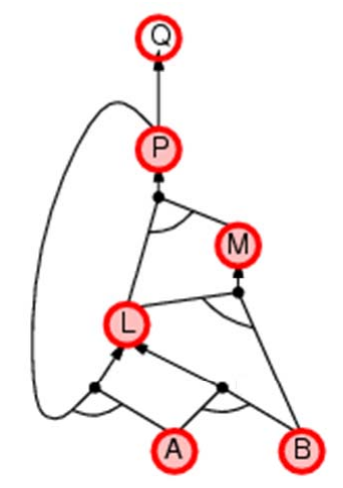
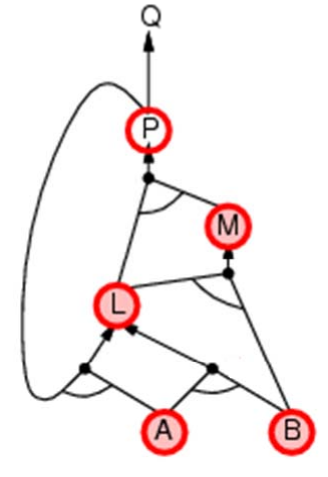
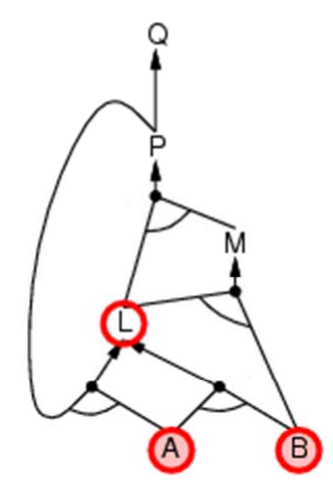
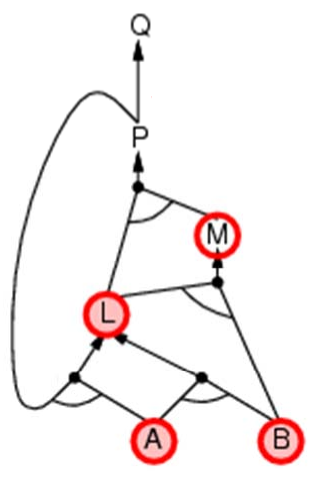
## Suy diễn tiến (Forward chaining)

* Với một tập các mệnh đề giả thiết (cơ sở tri thức) KB, cần suy ra

mệnh đề kết luận Q.

* Ý tưởng: Lặp lại 2 bước sau cho đến khi suy ra được kết luận.
  + Áp dụng các luật có mệnh đề giả thiết được thỏa mãn trong KB.
  + Bổ sung kết luận của các luật đó vào KB.

1. P ⇒ Q
2. L ∧ M ⇒ P
3. B ∧ L ⇒ M
4. A ∧ P ⇒ L
5. A ∧ B⇒ L
6. A
7. B



## Suy diễn lùi (Backward chaining)

* Ý tưởng: Quá trình suy diễn bắt đầu từ mệnh đề kết luận Q.
* Để chứng minh Q bằng tập mệnh đề (cơ sở tri thức) KB:
  + Kiểm tra xem Q đã được chứng minh (trong KB) chưa?
  + Nếu chưa, tiếp tục chứng minh tất cả các mệnh đề giả thiết của một luật nào đó (trong KB) có mệnh đề kết luận là Q.
* Tránh các vòng lặp.
  + Kiểm tra xem các mệnh đề mới đã có trong danh sách các mệnh đề cần chứng minh chưa? – Nếu rồi, thì không bổ sung (lại) nữa!
* Tránh việc chứng minh lặp lại đối với 1 mệnh đề.
  + Đã được chứng minh (trước đó) là đúng.
  + Đã được chứng minh (trước đó) là không thể thỏa mãn được (sai) trong KB.

|  |
| --- |
|  |

## Suy diễn tiến hay suy diễn lùi?

* Suy diễn tiến là quá trình dựa trên dữ liệu (data-driven).
  + Ví dụ: việc nhận dạng đối tượng, việc đưa ra quyết định.
* Suy diễn tiến có thể thực hiện nhiều bước suy diễn dư thừa – chẳng liên quan tới (cần thiết cho) mục tiêu cần chứng mình.
* Suy diễn lùi là quá trình hướng tới mục tiêu (goal-driven), phù hợp cho việc giải quyết vấn đề.
  + Ví dụ: Làm sao để giành được học bổng của 1 chương trình PhD?.

III. Khảo sát, đặc tả yêu cầu bài toán, giải quyết bài toán

Bài toán 1: Với một tập các mệnh đề giả thiết (cơ sở tri thức) KB, cần suy ra mệnh đề kết luận Q.

Giả sử có tập giả thiết KB.

p ∧ q

p ⇒ r

(q ∧ r) ⇒ s

Cần chứng minh định lý s.

Bài toán 2:bắt đầu từ mệnh đề kết luận Q

1. **Áp dụng suy diễn tiến, suy diễn lùi giải quyết bài toán**
   1. **Phương pháp tiếp cận:**

Áp dụng giải thuật suy diễn tiến và suy diễn lùi để giải quyết hai bài toán trên

**Đầu vào và đầu ra**

* Input:    tập KB (kb2.txt) và các định lí ở tập q2.txt
* Output: hiển thị trong tập output.txt, in ra YES nếu mệnh đề tương ứng ở q2.txt được suy ra từ tập KB(kb2.txt) và NO nếu ngược lại.

**Input**

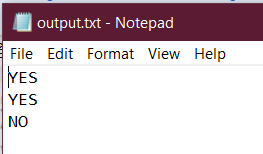
Trong cả 2 bài toán ta xét đầu vào của tập các mệnh đề giả thiết (cơ sở tri thức) KB và mệnh đề kết luận Q. Hai đầu vào này sẽ được lưu vào các file .txt như Fig.1

|  |  |
| --- | --- |
|  |  |

**Fig. 1**

**Output**

Nhiệm vụ của ta sẽ xem xét các mệnh đề trong Q có được suy diễn đến hay không. Nếu có đầu ra sẽ trả về YES, ngược lại sẽ trả về NO. Ví dụ trong bài toán trên ở Fig.1 ta có file output như sau:



**Fig.2**

Như trên **Fig.2** ta sẽ thấy mệnh đề **T** sẽ trả về YES, chứng tỏ mệnh đề đó đã được suy diễn tới, nhưng tự với mệnh đề trong Q, ta sẽ được file output.txt như **Fig.2.** Cụ thể chi tiết giải thuật thuật toán suy diễn tiến và suy diễn lùi như sau.

* 1. **Hướng giải quyết bài toán**

**Suy diễn tiến:**

Trong suy diễn tiến, đầu tiên ta khởi tạo 3 danh sách lần lượt lưu: mệnh đề có trong KB, các biểu thức có trong KB, và mệnh đề kết luận, ta sẽ gọi nó là kb\_list, clause\_list, và qlist. Thuật toán của ta sẽ chạy từng biểu thức trong clause\_list  và xem mệnh đề (target) được suy diễn của nó, nếu từ các mệnh đề có sẵn trong kb\_list có thể thỏa mãn được biểu thức đó ta thêm target vào kb\_list, ngược lại ta sẽ tiếp tục vòng lặp. Nếu một trong các biểu thức thỏa mãn và suy diễn ra các mệnh đề trong Q ta sẽ trả về YES và lưu vào file output.txt, nếu không thuật toán sẽ trả về NO.

**Suy diễn lùi:**

Ngược lại với thuật toán suy diễn tiến, thuật toán suy diễn lùi sẽ đọc từng mệnh đề q có trong Q (được lưu ở danh sách qlist) từ đó với mỗi mệnh đề ta mới xét ngược lên ở các biểu thức và mệnh đề có ở KB (cũng được lưu ở clause\_list, và kb\_list). Ở đây ta sẽ xét 2 điều kiện :

+ Nếu mệnh đề q có trong kb\_list -> trả về YES

+ Nếu không, mệnh đề q có được suy diễn từ các biểu thức trong clause\_list:

* ta xét các mệnh đề điều kiện của biểu thức đó, nếu thỏa mãn --> trả về YES

Nếu không, ta tiếp tục chạy đệ quy thuật toán suy diễn lùi với target mới là điều kiện chưa thỏa mãn.

* Nếu ít nhất một trong các mệnh đề điều kiện trong biểu thức đó trả về NO --> trả về NO

            + Trường hợp còn lại sẽ trả về NO

**Link tài liệu code java tham khảo**

https://github.com/siddhantj/PL-resolution--forward-and-backward-chaining?fbclid=IwAR11otMLzCgTqsFTmzpKSNZo8R1WDtjZaLS3Ch-YAfhHiPB3iw7xR5GYSfE