ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

TRƯƠNG NGỌC KHA

HUỲNH ĐĂNG KHOA

KHÓA LUẬN TỐT NGHIỆP

MỘT SỐ CẢI TIẾN CỦA TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH

GIẢNG VIÊN HƯỚNG DẪN

ThS. NGUYỄN ĐÌNH HIỂN

TP. HỒ CHÍ MINH, 2018

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

TRƯƠNG NGỌC KHA – 14520401

HUỲNH ĐĂNG KHOA - 14520422

KHÓA LUẬN TỐT NGHIỆP

MỘT SỐ CẢI TIẾN CỦA TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH

GIẢNG VIÊN HƯỚNG DẪN

ThS. NGUYỄN ĐÌNH HIỂN

TP. HỒ CHÍ MINH, 2018

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. ……………………………………… – Chủ tịch.
2. ……………………………………… – Thư ký.
3. ……………………………………… – Ủy viên.
4. ……………………………………… – Ủy viên.

NHẬN XÉT

(CỦA GIẢNG VIÊN HƯỚNG DẪN)

NHẬN XÉT

(CỦA GIẢNG VIÊN PHẢN BIỆN)

MỤC LỤC

Chương 1. TỔNG QUAN 16

1.1. Giới thiệu bài toán 16

1.1.1. Tổng quan về lập trình tập trả lời (Answer Set Programming – ASP) 16

1.1.2. Các phương pháp tính toán chương trình logic theo phương pháp đại số tuyến tính 18

1.2. Mục tiêu đề tài 20

1.3. Phương pháp thực hiện 20

Chương 2. TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH 21

2.1. Cơ sở lý thuyết 21

2.2. Chương trình logic dạng Horn 21

2.2.1. Biểu diễn ma trận cho chương trình horn 21

2.2.2. Thuật toán để tìm mô hình tối tiểu của chương trình logic dạng Horn 26

2.3. Chương trình logic dạng chuẩn 28

2.3.1. Chương trình logic dạng tuyển 28

2.3.1.1. Tensor biểu diễn chương trình logic dạng tuyển 28

2.3.1.2. Thuật toán để tìm mô hình tối thiểu của một chương trình dạng tuyển 32

2.3.2. Chương trình logic dạng chuẩn 34

Chương 3. PHƯƠNG PHÁP CÁI TIẾN TÍNH TOÁN CÁC MÔ HÌNH CỦA CÁC CHƯƠNG TRÌNH LOGIC 35

3.1. Chương trình dạng horn 35

3.1.1. Chương trình thoả điều kiện đơn (Singly-Define-condition) 35

3.1.2. Xây dựng chương trình thoả điều kiện đơn 37

3.1.3. Thuật toán tìm mô hình tối tiểu 38

3.1.4. Tính bằng ma trận con 39

3.2. Chương trình dạng chuẩn (Normal Program) 43

3.2.1. Tính toán mô hình ổn định 43

3.2.2. Thuật toán để tính toán mô hình ổn định 45

3.2.3. Tính toán bằng ma trận con 47

Chương 4. KẾT QUẢ THỬ NGHIỆM 50

4.1. So sánh với chương trình logic dạng horn 50

4.2. So sánh với chương trình logic dạng chuẩn 53

Chương 5. KẾT LUẬN 57

5.1. Kết quả đạt được 57

5.2. Hướng phát triển 57

DANH MỤC BIỂU ĐỒ

Biểu đồ 1. Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình xác định 52
Biểu đồ 2. Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình dạng chuẩn 55

DANH MỤC BẢNG

Bảng 1.Tỉ lệ của các luật trong *p* theo số lượng của các biến mệnh đề trong phần thân 50
Bảng 2. Kết quả thử nghiệm trên chương trình dạng Horn 51
Bảng 3. Tỉ lệ các luật trong *P* dựa trên số lượng các biến mệnh đề trong vế trái (Head) 53
Bảng 4. Kết quả thử nghiệm của chương trình dạng chuẩn 54

DANH MỤC TỪ VIẾT TẮT

- ASP: Answer Set Programming
- MD : Multiple Definitions
- SD: Singly Define
- FP: Fix Point
- MM: Minimal Model
- LM: Least Model

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời biết ơn sau sắc đến thầy Nguyễn Đình Hiển đã tận tình hướng dẫn chúng em suốt quá trình thực hiện khoá luận tốt nghiệp này. Nhờ sự tận tâm giúp đỡ nhiệt tình của thầy, chúng em mới có thể hoàn thành tốt khoá luận này.

Chúng em xin chân thành cãm ơn các thầy cô giáo Trường Đại Học Công Nghệ Thông Tin – Đại Học Quốc Gia Thành Phố Hồ Chí Minh trong suốt bốn năm học vừa qua đã tận tình giảng dạy và cung cấp cho chúng em những kiến thức quý báu.

Chúng em cũng đặt biệt cảm ơn những thầy cô trong Khoa Khoa Học Máy Tính đã tạo điều kiện và cung cấp mọi thứ cần thiết để em thực hiện khoá luận này thành công nhất.

Cuối cùng chúng em chúc các thầy cô dồi dào sức khoẻ, đạt được nhiều thành công trên con đường giảng dạy những thế hệ sau này. Sự tận tình và quan tâm của thầy cô luôn là động lực để những sinh viên như chúng em ngày một phát triển hơn.

TP. Hồ Chí Minh, Tháng 7 năm 2018

Trương Ngọc Kha

Huỳnh Đăng Khoa

TÓM TẮT KHÓA LUẬN

Chương trình logic là một mô hình biểu diễn tri thức dựa trên logic, và cung cấp các ngôn ngữ cho việc giải quyết vấn đề khai báo và lý luận đặc trưng. Trong nghiên cứu này, chúng tôi sẽ nghiên cứu phương pháp tính toán ngữ nghĩa của một chương trình logic dựa trên đại số tuyến tính. Các chương trình Horn logic và chương trình dạng chuẩn được biểu diễn bằng ma trận và mô hình tối tiểu cũng như mô hình ổn định của nó sẽ được xác định dựa trên các kỹ thuật tính toán trên ma trận. Bên cạnh đó, các chương trình logic dạng tuyển cũng được nghiên cứu và biểu diễn bằng các tensor. Thông qua đó, sử dụng các tensor này để tính toán các mô hình tối tiểu của chương trình logic dạng tuyển. Hơn nữa, các kỹ thuật tối ưu cho quá trình tính toán các mô hình này cũng được nghiên cứu và đề xuất. Kết quả phân tích và thử nghiệm trong thực tế cho thấy phương pháp tối ưu được đề xuất mang lại những hiệu quả nhất định.

Khoá luận được chia thành 5 chương với nội dung chi tiết các chương như sau:

- Chương 1. Tổng quan: Giới thiệu tổng quan về Answer Set Programming (ASP) là gì và sơ lược về các phương pháp tính toán chương trình logic theo phương pháp đại số tuyến tính. Từ đó, đưa ra hướng tiếp cận, mục tiêu cũng như phương pháp thực hiện của khoá luận này.
- Chương 2. Tính toán chương trình logic theo tiếp cận đại số tuyến tính: Chương này sẽ nói về cơ sở lý thuyết, cách biểu diễn trên ma trận và thuật toán tìm mồ hình tối tiểu của chương trình dạng Horn và dạng tuyển.
- Chương 3. Các phương pháp cải tiến tính toán của chương trình logic: Trình bày về các phương pháp cải tiến, cách biểu diễn ma trận theo một cách khác giúp giảm kích thước ma trận. Từ đó, tối ưu lại ma trận này
- Chương 4. Kết quả thử nghiệm: Tiến hành thực nghiệm các thuật toán tìm mô hình tối tiểu và các cải tiến cũng chúng để xem kết quả thực tế của chúng và cho thấy được độ hiểu quả của các phương pháp cải tiến

- Chương 5. Kết luận: Chương này sẽ tổng kết lại các kết quả đạt được của các thuật toán cải tiến và một số hướng phát triên khác sau này để cải thiện tốt hơn sau này

MỞ ĐẦU

Hiện nay, con người vẫn đang tìm kiếm những phương pháp giúp cho việc giải quyết các vấn đề tính toán trở nên dễ dàng hơn. Nếu ta chỉ xét trên phạm vi giải quyết các vấn đề tính toán trên máy tính thì Answer Set Programming (ASP) - một mô hình đơn giản và trực quan cho việc tính toán - chính là thứ đáp ứng yêu cầu đó.

Chương trình logic là một phương pháp để đặc tả và suy luận hình thức cho các vấn đề. Đại số tuyến tính cũng đã được ứng dụng nhiều trong tính toán khoa học. Hiện nay, trong lĩnh vực trí tuệ nhân tạo, việc tích hợp đại số tuyến tính và tính toán symbolic đang ngày càng được quan tâm hơn [14]. Đã có những nghiên cứu về việc biểu diễn các chương trình logic thông qua đại số tuyến tính.

Hiện nay, đã tồn tại những nghiên cứu để biến dổi một chương trình datalog thành một tập các phương trình ma trận [15]. Giải các phương trình ma trận này, ta có thể tìm được mô hình tối tiểu của chương trình. Cũng có những nghiên cứu về việc biễu diễn việc học khái niệm và quan hệ trong một hệ cơ sở tri thức dựa trên tiếp cận nơron nhúng [13]. Hay việc áp dụng mạng logic tensor (logic tensor network) vào logic thực (real logic). Ví dụ như bộ công cụ mạng tính toán (computation network toolkit) của Microsoft [12].

Ta cũng có thể biểu diễn một chương trình logic bằng phương pháp đại số tuyến tính. Như trong [16,17], ánh xạ tuyến tính và tensor được tác giả dùng để biểu diễn các vị từ, các quan hệ và các cơ sở logic. Lin đã trình bày một phương pháp tính toán đại số tuyến tính cho việc giải bài toán SAT (satisfiability problem) [4]. Bell và đồng sự của ông cũng đã sử dụng phương pháp quy hoạch nguyên hỗn hợp (mixed integer programming) để tính toán các mô hình tối tiểu và mô hình ổn định của một chương trình logic dạng chuẩn (normal logic program) [18]. Trong [1], các chương trình logic dạng Horn, dạng tuyển, dạng chuẩn cũng được biểu diễn bằng các tensor.

Trong khóa luận này, dựa trên những nghiên cứu trong [1], chúng tôi trình bày các thuật toán để xác định mô hình tối tiểu của chương trình dạng Horn cũng như mô hình ổn định của chương trình dạng chuẩn. Đồng thời, nghiên cứu và giới thiệu phương pháp khác để giảm kích thước của ma trận khi biểu diễn chương trình logic bằng ma trận.

Chúng em rất mong nhận được các nhận xét và ý kiến đóng góp từ phía thầy cô.

Chương 1. TỔNG QUAN

1.1. Giới thiệu bài toán

1.1.1. Tổng quan về lập trình tập trả lời (Answer Set Programming – ASP)

Answer Set Programming (ASP) [2] là một phương pháp mới nổi để mô hình hóa và giải quyết các vấn đề tìm kiếm và tối ưu hóa. Nó là sự kết hợp của một phương pháp đặc tả vấn đề dựa trên mô hình, một ngôn ngữ biểu diễn biểu cảm và những công cụ giải quyết hiệu quả. ASP giúp biểu diễn một cách trực quan và tự nhiên những kiến thức về những lĩnh vực hoặc vấn đề cụ thể, bao gồm cả những kiến thức không hoàn thiện, mặc định hay sở thích. Do khía cạnh khai báo mạnh mẽ của mình, ASP giúp đẩy nhanh việc tạo mẫu và phát triển phần mềm dùng để giải quyết các vấn đề tìm kiếm và tối ưu hóa cũng như tạo điều kiện cho các sửa đổi để tăng cường hiệu năng của chương trình .Về mặt nguyên tắt, không giống như truy vấn Prolog, có thể dẫn đến lặp vô hạn, quá trình tính toán bằng phương pháp ASP luôn kết thúc.

Thành phần của ASP bao gồm các phần tử (atoms), mệnh đề (literals), và luật (rules) [2]. Phần tử là các mệnh đề cơ bản có thể đúng hoặc sai, mệnh đề là các phần tử và dạng phủ định của chúng. Luật là một biểu thức có dạng như sau:

$$p \leftarrow q, \neg r.$$

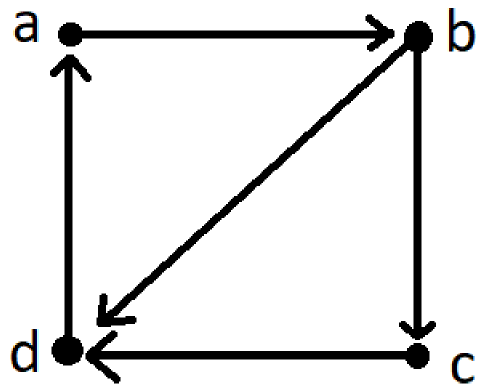
Phần bên trái ← gọi là head, phần bên phải gọi là body. Luật có thể không có body, chúng được gọ là fact và phần head được xem là đúng, ví dụ:

$$p \leftarrow .$$

Tập trả lời ngữ nghĩa của chương trình là nền tảng của ASP, tuy nhiên, việc thấu hiểu cách chương trình mã hóa các vấn đề tìm kiếm và những thể hiện của chúng cũng không kém phần quan trọng [2].

Chương trình là một tập hợp các luật. Để hiểu rõ hơn về ASP, ta xem xét ví dụ sau.

Ví dụ 1.1.1: Cho một đồ thị có hướng G = (V,E) như hình bên dưới. V là tập đỉnh còn E là tập cạnh của đồ thị. Tìm chu trình Hamilton của đồ thị G.



Ta có thể biểu diễn đồ thị G bằng tập các phần tử như sau:

D_{hc}(G) = {vtx (a), vtx (b), vtx (c), vtx (d)} ∪ {edge(a, b), edge(b, c), edge(c, d), edge(d, a), edge(b, d)}.

Khi một cạnh (a,b) được chọn cho chu trình Hamilton, ta kí hiệu là in(a,b). Để chỉ ra một cạnh bất kỳ (X,Y) được chọn cho chu trình Hamilton, ta sử dụng luật:

(HC1) {in (X, Y)} ← edge (X, Y).

Ta cũng quy định rằng không có cạnh nào bắt đầu và kết thúc ở cùng một đỉnh. , do đó, ta có 2 luật ràng buộc sau:

(HC2) ←in (V 2, V 1), in (V 3, V 1), V 2 ≠ V 3.

(HC3) ←in (V 1, V 2), in (V 1, V 3), V 2 ≠ V 3.

Để biểu diễn một đỉnh được dẫn từ một đỉnh khác, ta sử dụng luật:

(HC4) rchble(V, V)

(HC5) rchble(V 1, V 3) ← in(V 1, V 2), rchble(V 2, V 3).

Các luật (HC4) và (HC5) xác định chu trình đóng của quan hệ "in". Có nghĩa là tất cả các tập đỉnh (x,y) sao cho y có thể đi được từ x bằng 0 hay nhiều cạnh được xác định “in”. Các cặp cạnh được chọn tạo thành chu trình Hamilton khi và chỉ khi nó ở trong một chu trình đóng. Ta biểu diễn luật này như sau:

(HC6) ← vtx (V 1), vtx (V 2), not rchble(V 1, V 2).

Cho P_{hc} là một chương trình bao gồm các luật từ (HC1) – (HC6). Một tập cạnh H là chu trình Hamilton của đồ thị G khi vào chỉ khi :

H = {(x, y) | in(x, y) ∈ M} với M là answer set của $P_{hc} \cup D_{hc}(G)$.

Giả sử ta có một chương trình P và một tập các phần tử thuộc chương trình P gọi là M và ta không phát sinh được bất kì phần tử nào khác ngoài M từ chương trình P. Thì với các luật thuộc P mà có chứa mệnh đề ¬q ,với q thuộc M, thì không thể sử dụng được.

Ví dụ 1.1.2: Cho P1 là một chương trình bao gồm các luật sau đây:

a ← ¬b.

b ← ¬a.

Theo ví dụ trên, với tập M = {a} thì luật thứ 2 là b ← ¬a, do có chứa ¬a không thể sử dụng được. Do vậy, ta phát sinh được bất cứ mệnh đề nào ngoài tập M. Những tập như M được gọi là tập kết quả của P, hay còn gọi là mô hình của P [2].

Hiệu quả thực tế của ASP có thể thấy được qua việc nó đã được ứng dụng trong các lĩnh vực như: sinh học phân tử, hệ thống hỗ trợ quyết định cho tàu con thoi, hệ thống quản lý lực lượng lao động cho cảng biển Gioia Tauro, Reggio Calabria, Italy [2].

1.1.2. Các phương pháp tính toán chương trình logic theo phương pháp đại số tuyến tính

Chương trình dạng Horn là một chương trình logic không có mệnh đề dạng phủ định [1]. Cụ thể hơn, nó tập hữu hạng các luật có dạng:

$$h \leftarrow b_1 \wedge ... \wedge b_m \quad (m \geq 0) \qquad (1.1)$$

Trong đó, *h* và b_i là các biến mệnh đề.

Cho P là một chương trình dạng Horn, xét một tập I chỉ chứa các phần tử của P mang giá trị *true*. Với mọi luật có dạng (1.1) trong *P*, nếu $\{b_1 \ldots b_m\}$ suy ra h thuộc I và $\{b_1 \ldots b_m\}$ là tập con của I thì I được gọi là *mô hình* của P. Một chương trình P có thể có nhiều mô hình, khi đó, sẽ tồn tại một mô hình là con của tất cả các mô hình còn lại, ta gọi mô hình này là *mô hình tối tiểu (least model)* của P.

Chương trình dạng chuẩn (normal program) cũng giống như chương trình dạng Horn nhưng nó chứa luôn cả các mệnh đề phủ định (¬) [1]. Cụ thể hơn, một chương trình dạng chuẩn *P* là một tập hữu hạn các luật có dạng như sau:

$$h \leftarrow b_1 \wedge ... \wedge b_k \wedge \neg b_{k+1} \wedge ... \wedge \neg b_m \quad (m \geq 0) \qquad (1.2)$$

Với *h* và b_j là biến mệnh đề.

Theo [10], ta có định nghĩa về rút gọn (reduct) của một chương trình logic như sau. Cho một chương trình dạng chuẩn P, I là tập các phần tử thuộc P, rút gọn của P tương ứng với I chính là P sau khi ta loại bỏ các luật chứa mệnh đề phủ định của các phần tử thuộc I khỏi P và loại bỏ tiếp các mệnh đề phủ định còn lại trong phần body, tức là ta sẽ loại bỏ những luật chứa $\neg b_{k+1},\ldots, \neg b_m$ với $b_{k+1},\ldots,b_m$ thuộc I và loại bỏ $\neg b_{k+1},\ldots, \neg b_m$ còn lại khỏi phần body. Nếu mô hình tối tiểu của một rút gọn của P như vậy trùng với I thì I chính là mô hình ổn định của P. Với ví dụ 1.1.2 sau khi rút gọn của chương trình theo {a} thì ta được a ←. Mô hình tối tiểu của chương trình này cũng chính là {a}, do đó {a} là mô hình ổn định của chương trình ban đầu.

1.2. Mục tiêu đề tài

Về cơ bản, mục tiêu của đề tài là nghiên cứu phương pháp tính toán ngữ nghĩa của một chương trình logic theo phương pháp đại số tuyến tính. Các chương trình logic được biểu diễn bằng ma trận và mô hình tối tiểu hoặc mô hình ổn định của nó sẽ được xác định dựa trên các kỹ thuật tính toán trên ma trận. Cụ thể hơn, chúng tôi nghiên cứu phương pháp xây dựng chương trình logic dạng Horn, tính toán mô tối tiểu của chương trình và các thuật giải tối ưu của phương pháp đó. Tương tự với chương trình logic dạng chuẩn, tính toán các mô hình ổn định và các thuật giải tối ưu. Thử nghiệm để so sánh các kỹ thuật tối ưu được đề xuất với các phương pháp hiện hành để phân tích và xem xét hiệu quả của phương pháp cải tiến.

1.3. Phương pháp thực hiện

Đầu tiên chúng tôi nghiên cứu các phương pháp cấu trúc biểu diễn ma trận cũng như các tensor của chương trình logic đã được giới thiệu ở [1]. Sau đó, chúng tôi cũng nghiên cứu và đề xuất cải tiến của những phương pháp tính toán này bằng cấu trúc ma trận mới để biểu diễn lại chương trình đó mà biểu diễn này sẽ giãm được kích thước ma trận từ đó tối ưu lại các ma trận biểu diễn này. Không chỉ nghiên cứu và chứng minh tính hiệu quả của những cải tiến này về mặt lý thuyết, chúng tôi còn xây dựng chương trình thử nghiệm để so sánh kết quả đạt được của các phương pháp cải tiến được đề xuất này với những phương pháp đã được giới thiệu ở [1] để xem xét hiệu quả thực tế của chúng.

Chương 2. TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH

2.1. Cơ sở lý thuyết

Các định nghĩa trong mục này được trích từ [1, 9]. Tensor bậc k là kết quả của phép tính tích tensor của k vector. *Tensor bậc 1* là một vector $v \in \mathbb{R}^n$, *tensor bậc 2* là một ma trận $M \in \mathbb{R}^{m\times n}$, *tensor bậc 3* là một mảng ba chiều $T \in \mathbb{R}^{m\times n\times p}$ ($m,n,p\geq 1$). *Lớp* của một tensor bậc k là một mảng hai chiều của một tensor. Tensor bậc ba $T \in \mathbb{R}^{m\times n\times p}$ gồm các lớp bằng cách cố định một giá trị j ($1\leq j \leq p$). Lớp thứ j của một tensor bậc 3 T là một ma trận và được ký hiệu là $T_{::j}$.

Tích (2-mode) của một tensor $T \in \mathbb{R}^{m\times n\times p}$ và vector $v \in \mathbb{R}^n$ được ký hiệu là $T \bullet v$. Kết quả của tích này là một ma trận $m \times p$ có phần tử là $(T \bullet v)_{ik} = \sum_{j=1}^{n} x_{ijk} v_j$.

Khi $M \in \mathbb{R}^{m\times n}$ là một ma trận và $v \in \mathbb{R}^n$, tích $M \bullet v$ sẽ là một vector trong $\mathbb{R}^m$.

2.2. Chương trình logic dạng Horn

Xét ngôn ngữ L gồm có chứa một tập hữu hạn các biến mệnh đề và các phép toán logic $\neg$, $\wedge$, $\vee$ và $\leftarrow$. L có các ký hiệu $\top$ và $\bot$ đại diện cho các giá trị *true* và *false* một cách tương ứng. Cho một chương trình logic P, tập tất cả các biến mệnh đề xuất hiện trong P được gọi là cơ sở Herband của P, ký hiệu B_P. Giả định rằng $\{\top, \bot\} \subset B_P$.

2.2.1. Biểu diễn ma trận cho chương trình horn

Một chương trình logic P được là một chương trình (logic) *dạng Horn* khi và chỉ khi P là một tập hữu hạn các luật có dạng:

$$h \leftarrow b_1 \wedge ... \wedge b_m \quad (m \geq 0) \qquad (1)$$

Trong đó, h và b_i là các biến mệnh đề. Luật (1) là một *sự kiện* (*fact*) nếu "$h \leftarrow \top$" và (1) là một *ràng buộc* (constraint) nếu "$\bot \leftarrow b_1 \wedge \ldots \wedge b_m$". Khi đó, ta có thể viết lại chúng dưới dạng "$h \leftarrow$" và "$\leftarrow b_1 \wedge \ldots \wedge b_m$". Một chương trình được gọi là *xác định* (definite program) khi nó không chứa các ràng buộc.

Với luật r có dạng (1) trong chương trình dạng Horn P, đặt:

$$head(r) = h \text{ và } body(r) = \{b_1, \ldots, b_m\}.$$

Xét một tập I thỏa $\{\top\} \subseteq I \subseteq B_P$, $\bot \notin I$.

Khi đó, I được gọi là một *mô hình* của P nếu $\{b_1, \ldots, b_m\} \subseteq I$ suy ra $h \in I$ với mọi luật có dạng (1) trong P. Một mô hình I là *mô hình tối tiểu* (*least model*) của P nếu $I \subseteq J$ $\forall$ J là mô hình của P.

Ánh xạ $T_P : 2^{B_P} \to 2^{B_P}$ được định nghĩa như sau:

$$T_P(I) = \{h \mid h \leftarrow b_1 \wedge \ldots \wedge b_m \in P \text{ và } \{b_1, \ldots, b_m\} \subseteq I\}$$

Luỹ thừa của T_P được định nghĩa như sau:

$$T_P^{k+1}(I) = T_P\left(T_P^k(I)\right) \text{ và } T_P^0(I) = I \ (k \geq 0).$$

Cho $I \subseteq B_P$, khi đó sẽ có một điểm cố định $T_P^{n+1}(I) = T_P^n(I) \quad (n \geq 0)$.

Cho một chương trình P xác định, điểm cố định $T_P^n(\{\top\})$ cũng chính là mô hình tối tiểu của P. x

Định lý 2.2.1 [3]: Cho chương trình Horn P và ánh xạ T_P.

a) Đặt $M := \bigcap J$, với J là mô hình của P. Thì M là mô hình tối tiểu của P.

b) M là điểm cố định nhỏ nhất của T_P, nghĩa là $M = T_P^n(\{\top\})$.

Định nghĩa 2.2.1: (*MD-điều kiện*)

Chương trình dạng Horn P được gọi là thỏa *MD-điều kiện* khi và chỉ khi P thỏa điều kiện sau:

Với mọi luật r_1 và r_2 tthuộc P ($r_1 \neq r_2$):

$$head(r_1) = head(r_2) \text{ suy ra } |body(r_1)| \leq 1 \text{ và } |body(r_2)| \leq 1.$$

(gọi là *MD-điều kiện* (*multiple definitions*)).

Bổ đề 2.2.1: Cho chương trình dạng Horn P và M là mô hình tối tiểu của P. Khi đó, có một chương trình dạng Horn P' thoả MD-điều kiện, đồng thời:

P' có mô hình tối tiểu là M' và $M' \cap B_P = M$.

Chứng minh:

Đặt $S := \{r \mid r \in P,\ |body(r)| > 1\}$.

$P' = P$; $B_{P'} = B_P$;

Với mỗi $r = h_r \leftarrow b_{r1} \wedge \ldots \wedge b_{rm} \in S$, tạo một biến mệnh đề mới $c_r \notin B_P$ và hai luật mới: $h_r \leftarrow c_r$ và $c_r \leftarrow b_{r1} \wedge \ldots \wedge b_{rm}$.

$B_{P'} := B_P \cup \{c_r\}$.

$P' := P' \setminus \{r\} \cup \{h_r \leftarrow c_r ,\ c_r \leftarrow b_{r1} \wedge \ldots \wedge b_{rm}\}$.

Khi đó P' sẽ thoả MD-điều kiện và mô hình tối tiểu M' của P' thoả: $M' \cap B_P = M$.

Định nghĩa 2.2.2 [1]: *(ma trận biểu diễn chương trình Horn)*

Cho một chương trình dạng Horn P thoả MD-điều kiện và $B_P = \{b_1, \ldots, b_n\}$. Khi đó, P được biểu diễn bởi một ma trận $M_P \in \mathbb{R}^{n \times n}$ với mỗi phần tử a_{ij} $(1 \le i, j \le n)$ trong M_P có giá trị:

1. $a_{ij} = 1$ nếu $b_i = \top$ hoặc $b_j = \bot$.
2. $a_{ij_k} = \dfrac{1}{m}$ $(1 \le k \le m;\ 1 \le i, j_k \le n)$ nếu $b_i \leftarrow b_{j_1} \wedge \ldots \wedge b_{j_m}$ có trong P.
3. Các trường hợp khác, $a_{ij} = 0$.

Đặt hàng$_i(M_P) = (a_{i1}, a_{i2}, \ldots, a_{in}) \in \mathbb{R}^n$,

cột$_j(M_P) = (a_{1j}, a_{2j}, \ldots, a_{nj})^T$ $(1 \le i, j \le n)$

(v^T là một chuyển vị của vector $v \in \mathbb{R}^n$).

Định nghĩa 2.2.3 [1]: (*vector biểu diễn mô hình*)

Cho P là một chương trình dạng Horn và $B_P = \{b_1,\ldots, b_n\}$. thì mô hình I của P được biểu diễn bởi một vector $v = (a_1, \ldots, a_n)^T$ trong đó mỗi phần tử a_i biểu diễn giá trị thực của mệnh đề p_i sao cho $a_i = 1$ nếu $b_i \in I$ $(1 \le i \le n)$; và $a_i = 0$ trong các trường hợp còn lại.

Vector biểu diễn cho $I = \{\top\}$ is được ký hiệu là v_o.

Giả sử hai vector $v = (a_1,\ldots, a_n)^T$ và $w = (b_1,\ldots,b_n)^T$ lần lượt biễu diễn các mô hình $I \subseteq B_P$ và $J \subseteq B_P$ tương ứng.

Khi đó $v \le w$ nếu $a_i \le b_i \;\; \forall\, 1 \le i \le n$.

Ví dụ 2.2.1:

a) Xét $P1 = \{p \leftarrow q, p \leftarrow r, q \leftarrow r \wedge s, r \leftarrow ,\ \leftarrow q\}$ với $B_{P1} = \{p, q, r, s, \top, \bot\}$ thì $M_{P1} \in \mathbb{R}^{6\times 6}$ là ma trận vuông.

Hàng 1 "011001" tượng trưng $p \leftarrow q, p \leftarrow r, p \leftarrow \bot$.

Hàng 2 "00½ ½01" biểu diễn các luật $q \leftarrow r \wedge s, q \leftarrow \bot$.

Hàng 3 "000011" biểu diễn các luật $r \leftarrow, r \leftarrow \bot$.

Hàng 4 "000001" biểu diễn luật $s \leftarrow \bot$.

Hàng 5 "111111" biểu diễn các luật $\top \leftarrow p, \top \leftarrow q, \top \leftarrow r, \top \leftarrow s, \top \leftarrow \top, \top \leftarrow \bot$.

Hàng 6 "010001" biểu diễn các luật $\bot \leftarrow q, \bot \leftarrow \bot$.

$$
\begin{array}{c}
\begin{array}{cccccc} p & q & r & s & \top & \bot \end{array} \\
\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & S & S & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}
\begin{array}{c} p \\ q \\ r \\ s \\ \top \\ \bot \end{array}
\end{array}
$$

b) Xét $P2 = \{p \leftarrow t, t \leftarrow q \wedge r, p \leftarrow u, u \leftarrow s \wedge q, s \leftarrow r, r \leftarrow q\}$ với $B_{P2} = \{p, q, r, s, t, u\}$ thì $M_{P2} \in \mathbb{R}^{6\times 6}$ là ma trận (vuông).

Để biểu diễn đơn giản hơn, trong ví dụ này chúng ta bỏ qua $\top$ và $\bot$.

Hàng 1 “000011” biểu diễn các luật $p \leftarrow t$, $p \leftarrow u$.

Hàng 3 “010000” biểu diễn luật $r \leftarrow q$.

Hàng 4 “001000” biểu diễn luật $s \leftarrow r$.

Hàng 5 “0 ½ ½ 000” biểu diễn luật $t \leftarrow q \wedge r$.

Hàng 6 “0 ½ 0 ½00” biểu diễn luật u $\leftarrow s \wedge q$.

$$\begin{array}{cc} \begin{matrix} p & q & r & s & t & u \end{matrix} & \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & S & S & 0 & 0 & 0 \\ 0 & S & 0 & S & 0 & 0 \end{pmatrix} & \begin{matrix} p \\ q \\ r \\ s \\ t \\ u \end{matrix} \end{array}$$

Giả sử $M_P \in \mathbb{R}^{n\times n}$ là ma trận biểu diễn cho một chương trình dạng Horn P. Gọi $v \in \mathbb{R}^n$ là vector biểu diễn cho một mô hình $I \subseteq B_P$, khi đó $M_P \bullet v = (a_1,\dots, a_n)^T$. Biến đổi $M_P \bullet v$ thành một vector $w = (a_1',\dots, a_n')^T$ trong đó $a_i' = 1$ $(1 \le i \le n)$ nếu $a_i \ge 1$; và $a_i' = 0$ nếu $a_i < 1$. Ký hiệu: $w = \theta(M_P v)$.

Định lý 2.2.2 [1]: Cho P là một chương trình xác định thoả MD-điều kiện và P có ma trận biểu diễn là $M_P \in \mathbb{R}^{n\times n}$. Gọi $v \in \mathbb{R}^n$ là vector biểu diễn cho một mô hình $I \subseteq B_P$.

Khi đó, $w \in \mathbb{R}^n$ là một vector biểu diễn cho $J = T_P(I)$ khi và chỉ khi $w = \theta(M_P v)$.

Chứng minh:

Cho $M_P = (a_{ij})$ và $v = (y_1,\dots,y_n)^T$ khi đó $w = \theta(M_P v) = (x_1,\dots,x_n)^T$ với $x_k = a_{k1}y_1 + \dots + a_{kn}y_n$.

+ Giả sử $w = \theta(M_P v) = (x_1',\dots,x_n')^T$ với $x_k' = 1 \Leftrightarrow x_k \ge 1$.

Ta có: $x_k = a_{k1}y_1 + \dots + a_{kn}y_n \ge 1$

Nếu $hàng_j(M_p) = T$ thì $a_{kj} = 1$.

Cho $\{b_1,\dots,b_m\} \subseteq \{a_{k1},\dots,a_{kn}\}$ sao cho $b_i \neq 0$. Ta có 2 trường hợp:

i/ $b_i = 1$ $(1 \le i \le m)$ và $b_1 y_{j1} + \dots + b_m y_{jm} \ge 1$

Khi có tồn tại luật: $p_k \leftarrow p_i$ trong P sao cho $p_i = cột_j(M_p)$ để $b_i = a_{kj}$ $(1 \le i \le m)$ và $\{p_1,\dots,p_m\} \subseteq I$ nghĩa là $p_k \in T_p(I)$ khi $p_k = hàng_k(w)$.

ii/ $b_i = 1/m$ $(1 \le i \le m)$ và $b_1 y_{j1} + \dots + b_m y_{jm} = 1$

Khi có tồn tại luật: $p_k \leftarrow p_1 \wedge ... \wedge p_m$ trong P sao cho $p_i = cột_j(M_p)$ để $b_i = a_{kj}$ $(1 \leq i \leq m)$ và $\{p_1, ..., p_m\} \subseteq I$ nghĩa là $p_k \in T_p(I)$ khi $p_k = hàng_k(w)$.

+ Giả sử $J = T_p(I)$:

Định nghĩa $w = (x_1', ..., x_n')^T$ biểu diễn $J = T_p(I)$.

Ta có: $w = \theta(M_P v) = (x_1, ..., x_n)^T$ là một vector sao cho:

$$x_k \geq 1 \Leftrightarrow hàng_k(w = \theta(M_P v)) = T \text{ hoặc } hàng_k(w = \theta(M_P v)) \in T_p(I)$$

Khi đó w sao cho: $x_k' = 1 \Leftrightarrow 1$ trong $w = \theta(M_P v)$

Vì vậy $w = \theta(M_P v)$

Cho ma trận biểu diễn $M_P \in \mathbb{R}^{n \times n}$ và một vector $v \in \mathbb{R}^n$. Định nghĩa:

$\theta(M_P v_{k+1}) = \theta(M_P (M_P v_k)$ và $\theta(M_P v_1) = \theta(M_P v)$ $(k \geq 1)$.

Khi $\theta(M_P v_{k+1}) = \theta(M_P v_k)$ với $k \geq 1$, đặt: $FP(M_P v) = \theta(M_P v_k)$.

Theo [1], ta có: $m \in \mathbb{R}^n$ là một vector biểu diễn cho mô hình tối tiểu cua P khi và chỉ khi $m = FP(M_P v_o)$ với v_o là vector biểu diễn $I = \{\top\}$.

2.2.2. Thuật toán để tìm mô hình tối tiểu của chương trình logic dạng Horn

Dựa trên định lý 3.2, chúng ta có thuật toán để tìm mô hình tối tiểu của một chương trình *P* thoả mãn MD-điều kiện. Không mất tính tổng quát, ta chỉ cần xét *P* là một chương trình xác định (không chứa các ràng buộc):

Input: *P* là một chương trình xác định thoả mãn MD-điều kiện và *F* là một tập các sự kiện trong *P*.

Output: Vector *u* biểu diễn mô hình tối tiểu của *P*.

Thuật toán 2.2.1:

Bước 1: *Tạo ma trận* $M_P = (a_{ij})_{1 \leq i,j \leq n}$ *biểu diễn chương trình P*	**Bước 3:** *Mô hình tối tiểu của P* $v := v_o$

for i from 1 to n do
for (luật $r \in P$) do
if $b_i \in head(r)$ then
$m := 1/\|body(r)\|$
for j from 1 to n do
if $b_j \in body(r)$ then
$a_{ij} = m$
end do; # $r \in P$
end do;
Bước 2: *Tạo vector*
$v_o = (v_1, \ldots v_n)$ *biểu diễn tập F*
for i from 1 to n do
if $b_i \in F$ then $v_i = 1$
else $v_i = 0$
end do;
	$u{:=}M_P \bullet v$
	$u := u + v_o$
	for i from 1 to n do
	#Biến đổi $u = \theta(M_P v)$
	if $u[i] \geq 1$ then $u[i] := 1$
	else $u[i]{:=}0$;
	end do;
	while $u \neq v$ do
	$v := v_o$
	$u{:=}M_P \bullet v$
	$u := u + v_o$
	for i from 1 to n do
	#Biến đổi $u = \theta(M_P v)$
	if $u[i] \geq 1$ then $u[i] := 1$
	else $u[i]{:=}0$;
	end do; #while
	return u;

Định lý 2.2.3: Cho P là một chương trình xác định và ánh xạ T_P. Gọi $\{u_k\}_{k\geq 0} \subseteq 2^{B_P}$ sao cho:

$$\begin{cases} u_0 = \varnothing \\ u_{k+1} = T_P(u_k) \quad \forall\ k \geq 0 \end{cases}$$

Đặt:

$$M = \bigcap_{J \text{ là một model của P}} J\,.$$

Khi đó, ta có:

a) $\forall\ k \geq 0$, $u_k \subseteq T_P(u_k) = u_{k+1}$.

b) $\exists n_o \in \mathbb{N}:\ u_{n_o} = T_P(u_{n_o})$ và $n_o \leq\ |B_P| = n$.

c) $u_{n_o} = M$.

Chứng minh.

a) Ta có thể dễ dàng chứng minh bằng phương pháp quy nạp.

b) Từ (a) ta có: $u_k \subseteq T_P(u_k) = u_{k+1}\ \forall\ k \geq 0$, và B_P có n phần tử

$$\Rightarrow \exists n_o \in \mathbb{N}:\ u_{n_o+1} = u_{n_o} \text{ và } n_o \leq n.$$

c) Kết quả này được suy ra từ định lý 2.2.1.

Trong thuật toán 2.2.1, độ phức tạp của $\theta(M_P v)$ là $\mathbf{O(n^2)}$. Theo định lý 2.2.3, số lần thực hiện $\theta(M_P v)$ nhiều nhất là ***n*** lần. Như vậy, độ phức tạp của bước 3 là $\mathbf{O(n^3)}$.

2.3. Chương trình logic dạng chuẩn

2.3.1. Chương trình logic dạng tuyển

Một chương trình logic P là một *chương trình* (*logic*) *dạng tuyển (disjunctive (logic) program)* khi và chỉ khi P là một tập hữu hạn các luật có dạng:

$$h_1 \vee \ldots \vee h_l \leftarrow b_1 \wedge \ldots \wedge b_m \quad (l \geq 1, m \geq 0) \qquad (2)$$

Trong đó h_i và b_j là các biến mệnh đề. Luật dạng (2) được gọi là một *luật tuyển* nếu $l > 1$, là *sự kiện* nếu vế phải của (2) là $\top$ và gọi là một *ràng buộc* nếu vế trái của (2) là $\perp$. Khi đó, một sự kiện dạng tuyển được viết đơn giản như sau :

$$h_1 \vee \ldots \vee h_l \leftarrow.$$

2.3.1.1. Tensor biểu diễn chương trình logic dạng tuyển

Cho luật r trong một chương trình dạng tuyển P, đặt:

$head(r) = \{h_1, \ldots, h_l\}$ và $body(r) = \{b_1, \ldots, b_m\}$.

Một tập I thỏa điều kiện $\{\top\} \subseteq I \subseteq B_P$ được gọi là một mô hình của một chương trình dạng tuyển P nếu $body(r) \subseteq I$ suy ra $head(r) \cap I \neq \varnothing\ \ \forall\ r \in P$ và $\perp \notin I$.

Một mô hình I là một mô hình tối thiểu (minimal model) của P nếu không tồn tại mô hình J của P mà $J \subset I$.

Định nghĩa 2.3.1: (*MD-điều kiện cho chương trình dạng tuyển*)

Chương trình dạng tuyển P được gọi là thỏa MD-điều kiện nếu nó thỏa những điều kiện sau:

$$\forall\, r_1 \in P, r_2 \in P, r_1 \neq r_2:$$

$head(r_1) \cap head(r_2) \neq \varnothing$ suy ra $|body(r_1)| \leq 1$ và $|body(r_2)| \leq 1$.

(gọi là *MD-điều kiện* (*multiple definitions*)).

Định nghĩa này trùng với định nghĩa 2.2.1 nếu P là một chương trình logic dạng Horn.

Bổ đề 2.3.1: Cho một chương trình logic dạng tuyển P, khi đó, có một chương trình logic dạng tuyển P' thoả MD-điều kiện, đồng thời:

P có một mô hình tối thiểu M khi và chỉ khi P' có mô hình tối thiểu là M' sao cho $M' \cap B_P = M$.

Chứng minh [1]: Đặt $S := \{r \mid r \in P,\ |body(r)| > 1\}$.

$P' := P;\quad B_{P'} := B_P;$

Với mỗi $r = h_{r1} \vee \ldots \vee h_{rl} \leftarrow b_{r1} \wedge \ldots \wedge b_{rm} \in S$ ($l \geq 1$, $m > 1$), tạo một biến mới $c_r \notin B_P$ và hai luật mới: $h_{r1} \vee \ldots \vee h_{rl} \leftarrow c_r$ và $c_r \leftarrow b_{r1} \wedge \ldots \wedge b_{rm}$.

$B_{P'} := B_P \cup \{c_r\}$.

$P' := P' \setminus \{r\} \cup \{ h_{r1} \vee \ldots \vee h_{rl} \leftarrow c_r,\ c_r \leftarrow b_{r1} \wedge \ldots \wedge b_{rm}\}$.

Khi đó P' thỏa MD-điều kiện, đồng thời:

P có một mô hình tối thiểu M khi và chỉ khi P' có mô hình tối thiểu là M' sao cho $M' \cap B_P = M$.

Cho P là một chương trình dạng tuyển, *chương trình tách* (*split program*) của P là một chương trình Horn được sinh ra từ P bằng cách thay thế mỗi luật tuyển dạng (2) trong P bằng một luật Horn:

$$h_i \leftarrow b_1 \wedge \ldots \wedge b_m \quad (1 \leq i \leq l)$$

Theo định nghĩa, *P* có nhiều chương trình tách. Khi *P* có *k* chương trình split, chúng sẽ được ký hiệu là $SP_1,\ldots, SP_k$.

Gọi T là tập các mô hình, khi đó phần tử tối thiểu của tập T được định nghĩa như sau:

min(T) = { $I \mid I \in$ T và không tồn tại $J \in$ T sao cho $J \subset I$ }.

Mệnh đề 2.3.1 [1]: Cho *P* là một chương trình dạng tuyển và $SP_1,\ldots, SP_k$ là các chương trình tách từ P. Gọi *LM* là tập hợp các mô hình tối tiểu (least model) của $SP_1,\ldots, SP_k$. Khi đó min(*LM*) trùng với tập *MM* là tập các mô hình tối thiểu (minimal model) của *P*.

Chứng minh:

Cho M là mô hình tối tiểu của chương trình tách SPj.

Khi đó với mỗi luật r: $h_i \leftarrow b_1 \wedge \ldots \wedge b_m$ trong SPj, tồn tai một luật r': $h_1 \vee \ldots \vee h_l \neg b_1 \wedge \ldots \wedge b_m$ trong P sao cho $h_i \in$ head(r'). Vì M thoả r, nên M thoả r'. Vậy, M là một mô hình của P.

Khi đó tập tối thiểu của M trong min(LM) là một mô hình tối thiểu (minimal model) của P.

Như vậy, min(LM)⊆MM.

Ngược lại, cho M∈MM.

Khi đó với mỗi luật r: $h_1 \vee \ldots \vee h_l \neg b_1 \wedge \ldots \wedge b_m$ trong P, $\{b_1, \ldots, b_m\} \subseteq$ M hàm ý $h_i \in$ M với i ($1 \leq i \leq l$).

Trong trường hợp này, tồn tại một chương trình tách SPj của P trong đó r được thay bằng $h_i \leftarrow b_1 \wedge \ldots \wedge b_m$.

Khi đó M là mô hình tối tiểu của SPj

Vì M là một mô hình tối thiểu trong LM, MM ⊆ min(LM).

Định nghĩa 2.3.2: (*tensor biểu diễn chương trình logic dạng tuyển*)

Cho *P* là một chương trình logic dạng tuyển thỏa MD-điều kiện. Giả sử rằng *P* được tách thành các chương trình dạng Horn $SP_1,\ldots, SP_k$ ($k \geq 1$) và $B_P = \{b_1, \ldots, b_n\}$. Khi đó *P* được biễu diễn by một tensor bậc ba $U^P \in \mathbb{R}^{n \times n \times k}$ như sau:

1. Mỗi lớp $U_{::h}$ $(1 \leq h \leq k)$ của U^P là một ma trận $M_h \in \mathbb{R}^{n\times n}$ biểu diễn chương trình Horn SP_h.

2. Mỗi ma trận $M_h \in \mathbb{R}^{n\times n}$ có một phần tử a_{ij} $(1 \leq i, j \leq n)$:

a) $a_{ij} = 1$ nếu $b_i = \top$ hoặc $b_j = \bot$

b) $a_{ij_k} = \dfrac{1}{m}$ $(1 \leq k \leq m;\ 1 \leq i, j_k \leq n)$

nếu $b_i \leftarrow b_{j_1} \wedge \ldots \wedge b_{j_m}$ thuộc P

c) Các trường hợp còn lại, $a_{ij} = 0$.

Ví dụ 2.3.1: Xét chương trình dạng tuyển P như sau: $P = \{p \vee r \leftarrow s, q \vee r \leftarrow, s \leftarrow\}$, $B_P = \{p, q, r, s, \top\}$, khi đó P được tách thành 4 chương trình dạng Horn:

$SP_1 = \{p \leftarrow s, q \leftarrow, s \leftarrow\}$,

$SP_2 = \{p \leftarrow s, r \leftarrow, s \leftarrow\}$,

$SP_3 = \{r \leftarrow s, q \leftarrow, s \leftarrow\}$,

$SP_4 = \{r \leftarrow s, r \leftarrow, s \leftarrow\}$.

Chương trình P được biểu diễn bởi tensor bậc ba $U^P \in \mathbb{R}^{5\times5\times4}$:

$$U_{::1} = \begin{array}{c} \begin{array}{ccccc} p & q & r & s & \top \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ p \\ q \\ r \\ s \\ \top \end{array} \qquad U_{::2} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$U_{::3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad U_{::4} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Gọi $U^P \in \mathbb{R}^{n\times n\times k}$ là một tensor bậc ba, $v \in \mathbb{R}^n$ là vector biểu diễn mô hình $I \subseteq B_P$, tích (2-mode) $U^P \bullet v$ là một ma trận $(a_{ij}) \in \mathbb{R}^{n\times k}$. Ta chuyển đổi $U^P \bullet v$ thành một ma trận $W = (a'_{ij}) \in \mathbb{R}^{n\times k}$ với $a'_{ij} = 1$ ($1 \le i \le n$, $1 \le j \le k$) nếu $a_{ij} \ge 1$; và $a'_{ij} = 0$ nếu $a_{ij}<1$. Khi đó, ta ký hiệu $W = \theta(U^P v)$.

Cho một tensor $U^P \in \mathbb{R}^{n\times n\times k}$ và vector $v \in \mathbb{R}^n$, định nghĩa:

$\theta(U^P v_{m+1}) = \theta(U^P(U^P v_m))$ và $\theta(U^P v_1) = \theta(U^P v_0)$ $(m \ge 1)$.

Khi $\theta(U^P v_{m+1}) = \theta(U^P v_m)$ với $m \ge 1$, đặt: $FP(U^P v) = \theta(U^P v_m)$.

Định lý 2.3.1 [1]: Cho P là một chương trình dạng tuyển không thỏa MD-điều kiện và $U^P \in \in \mathbb{R}^{n\times n\times k}$ là tensor biểu diễn cho chương trình P. Đặt $M_P = FP(U^P \bullet v_o)$ và v_o là vector biểu diễn I = {⊤}.

Khi đó, mỗi cột vector $m \in \mathbb{R}^n$ trong M_P biểu diễn một mô hình tối thiểu của P khi và chỉ khi m vector nhỏ nhất trong tất cả các cột vector trong M_P (quan hệ thứ tự giữa các vector đã được định nghĩa trong định nghĩa 2.2.3).

2.3.1.2. Thuật toán để tìm mô hình tối thiểu của một chương trình dạng tuyển

Input: P là một chương trình dạng tuyển thoả mãn MD-điều kiện và F là một tập các sự kiện trong P. Giả sử rằng P không có các ràng buộc.

Output: Tập MM các mô hình tối thiểu của P.

Thuật toán 2.3.1:

Bước 1: Tách chương trình P thành k chương trình Horn: $SP_1, SP_2,\ldots, SP_k$
 	
Bước 2: *Tạo tensor bậc ba $U^P \in \mathbb{R}^{n\times n\times k}$ để biểu diễn P*
for h from 1 to k do
	for i from 1 to n do
	# Biến đổi $M' = \theta(U^P v_F)$
	for j from 1 to k do
	if $M'[i, j] \ge 1$ then
	$M'[i, j] := 1$
	else $M'[i, j] := 0$;

<table>
<tr>
<td>

$U_{::h} := (a_{ij})_{1\le i,j\le n}$

for i from 1 to n do

  for (luật $r \in P$) do

    if $b_i \in head(r)$ then

      $m := 1/|body(r)|$

      for j from 1 to n do

        if $b_j \in body(r)$ then

          $a_{ij} = m$

    end if; # $b_i \in head(r)$

  end do; # r in P

end do;

end do;

Bước 3: *Tìm ma trận $M \in \mathbb{R}^{n\times k}$ trong đó các cột của nó biểu diễn mô hình tối tiểu của SP_h (h =1…k)*

3.1 Tạo vector $v_F = (v_1,\ldots v_n)$ biểu diễn tập sự kiện F.

3.2 Tạo ma trận $M_o \in \mathbb{R}^{n\times k}$ với $col_i(M_o) = v_F$ ($i = 1\ldots k$)

3.3 $M := U^P \bullet v_F$

$M := M + M_o$

$M' := M$

</td>
<td>

end do;

while $M \neq M'$ do

  $M := M'$

  $M' := U^P \bullet M$

  $M' := M + M'$

  for i from 1 to n do

  #Biến đổi $M' = \theta(U^P M)$

   for j from 1 to k do

     if $M'[i, j] \geq 1$ then

       $M'[i, j] := 1$

     else $M'[i, j] := 0$;

  end do;

end do; #while

Bước 4: *Tìm tập MM các vector biểu diễn các mô hình tối thiểu của P*

$MM := \varnothing$;

for i from 1 to k do

  $v_i := col_i(M)$;

  for j from 1 to k do

    $v_j := col_j(M)$;

    if ($v_j \leq v_i$) and ($v_j \neq v_i$) then break;

  end do;

  if $j > k$ then

    $MM := MM \cup \{v_i\}$;

end do;

</td>
</tr>
</table>

	return *MM*;

Trong thuật toán 2.3.1, độ phức tạp của phép tính $U^P \bullet M$ là $\mathbf{O}(\boldsymbol{k}^2 \times \boldsymbol{n}^2)$. Theo định lý 2.2.3, số lần thực hiện $\theta(U^P M)$ nhiều nhất là $\boldsymbol{n}$ lần. Như vậy, độ phức tạp của bước 2.2.3 là $(\boldsymbol{k}^2 \times \boldsymbol{n}^3)$.

2.3.2. Chương trình logic dạng chuẩn

Một *chương trình dạng chuẩn P* là một tập hữu hạn các *luật* có dạng như sau:

$$h \leftarrow b_1 \wedge \ldots \wedge b_k \wedge \neg b_{k+1} \wedge \ldots \wedge \neg b_m \quad (m \geq 0)$$

Chương trình dạng chuẩn có thể được chuyển đổi thành chương trình dạng tuyển (2) (mục 2.3.1) và sau đó giải mã chúng ở dạng ma trận cách sử dụng tensor bậc ba. Đầu tiên ta sẽ chuyển đổi chương trình dạng chuẩn thành chương trình xác định và sau đó giải mã chúng ở dạng ma trận như trong Mục 3.1.

Chương 3. PHƯƠNG PHÁP CÁI TIẾN TÍNH TOÁN CÁC MÔ HÌNH CỦA CÁC CHƯƠNG TRÌNH LOGIC

3.1. Chương trình dạng horn

Xét ngôn ngữ L gồm có chứa một tập hữu hạn các biến mệnh đề và các phép toán logic $\neg$, $\wedge$, $\vee$ và $\leftarrow$. Cho một chương trình logic *P*, tập tất cả các biến mệnh đề xuất hiện trong *P* được gọi là cơ sở Herband của P, ký hiệu B_P.

Sử dụng điều kiện MD thì quá trình chuyển đổi chương trình logic *P* dạng horn sang chương trình logic *P'* dạng horn thoả điều kiện MD thì sinh ra rất là nhiều các biến mới cũng như là nhiều các luật mới theo **Bổ Đề 2.3.1**. Nên trong phần này, chúng tôi thay đổi điều kiện MD thành một điều kiện mới mà quá trình đó làm giảm số lượng biến mới và các luật mới được sinh ra. Chúng tôi gọi nó là chương trình SD (Singly-Define) hay chương trình đơn

3.1.1. Chương trình thoả điều kiện đơn (Singly-Define-condition)

Đầu tiên chúng ta xét một lớp con của chương trình dạng Horn, được gọi là chương trình đơn.

Định nghĩa 3.1.1: (chương trình đơn) một chương trình *P* dạng Horn được gọi là *chương trình đơn* (hoặc SD-*program*) nếu $head(r_1) \neq head(r_2)$ với mọi luật r_1 và r_2 thuộc *P* ($r_1 \neq r_2$).

Định nghĩa 3.1.2 [1]: **(vector biểu diễn)** Cho *P* là một chương trình dạng Horn và $B_P = \{p_1, \ldots, p_n\}$. Thì mô hình *I* của *P* được biểu diễn bởi một vector $v = (a_1, \ldots, a_n)^T$ trong đó mỗi phần tử a_i biểu diễn giá trị thực của mệnh đề p_i sao cho $a_i = 1$ nếu $p_i \in I$ ($1 \leq i \leq n$); các trường hợp khác, $a_i = 0$. Chúng ta biết hàng$_i(v) = p_i$. Xét $v = (a_1, \ldots, a_n)^T \in \mathbb{R}^n$, $v[i]$ là phần tử thứ i^{th} của v ($1 \leq i \leq n$) và $v[1 \ldots k]$ là một vector $(a_1, \ldots, a_k)^T \in \mathbb{R}^k$ ($k \leq n$).

Định nghĩa 3.1.3: (ma trận biểu diễn của chương trình thoả điều kiện đơn)[1] cho P là một chương trình đơn và $B_P = \{p_1,..., p_n\}$. Khi đó, P được biểu diễn bởi một ma trận $M_P \in \mathbb{R}^{n\times n}$ với mỗi phần tử a_{ij} $(1 \le i, j \le n)$ trong M_P có giá trị:

1. $a_{ij_k} = \dfrac{1}{m}$ $(1 \le k \le m;\ 1 \le i, j_k \le n)$ nếu $p_i \leftarrow p_{j_1} \wedge ... \wedge p_{j_m} \in P$.
2. $a_{ii} = 1$ nếu $p_i \leftarrow$ có trong P.
3. $a_{ij} = 0$, các trước hợp khác.

M_P được gọi là một *ma trận biểu diễn*. chúng ta viết hàng$_i(M_P)= p_i$ và cột$_j(M_P) = p_j$ $(1 \le i, j \le n)$.

Định nghĩa 3.1.4: **(vector khởi tạo)** cho P là một chương trình dạng Horn và $B_P = \{p_1,..., p_n\}$. ta có *vector khởi tạo* $v_o = (a_1, . . . , a_n)^T$ vậy $a_i = 1$ nếu hàng$_i(v_o) = p_i$ và sự kiện $p_i \leftarrow$ trong P; trường hợp khác, $a_i = 0$.

Cho một ma trận biểu diễn $M_P \in \mathbb{R}^{n\times n}$ và một vector khởi tạo $v_o \in \mathbb{R}^n$, xác định:

$$v_1 = \theta(M_P v_o) \quad \text{và} \quad v_{k+1} = \theta(M_P v_k) \quad (k \ge 1).$$

Tồn tại $v_{k+1} = v_k$ với $k \ge 1$. Khi $v_{k+1} = v_k$, ta viết: $v_k = FP(M_P v_o)$.

Định lý 3.1.1: Cho P là một chương trình đơn và $M_P \in \mathbb{R}^{n\times n}$ là ma trận biểu diễn. ta có $m \in \mathbb{R}^n$ là một vector biểu diễn mô hình tối tiểu của P khi và chỉ khi $m = FP(M_P v_o)$ khi đó v_o là vector khởi tạo của P.

Ví dụ 3.1.1: Xét chương trình $P = \{p \leftarrow q,\ q \leftarrow p \wedge r,\ r \leftarrow s,\ s \leftarrow\}$ với $B_P = \{p, q, r, s\}$ vậy ma trận biểu diễn của nó $M_P \in \mathbb{R}^{4\times 4}$ là một ma trận (vuông). Vector khởi tạo của P là $v_o = (0\ 0\ 0\ 1)^T$.

ta có, $v_1 = \theta(M_P v_0) = (0\ 0\ 1\ 1)^T$ và $v_2 = \theta(M_P v_1) = (0\ 0\ 1\ 1)^T = v_1$.

$$\begin{array}{c c} \begin{matrix} p & q & r & s \end{matrix} & \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ S & 0 & S & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} p \\ q \\ r \\ s \end{matrix} \end{array}$$

---

[1] Trong [1], sự kiện được biểu diễn bằng cách "$p_i \leftarrow \top$" và được thể hiện trên ma trận bằng cách $a_{ij} = 1$ khi đó hàng$_i(M_P)= p_i$ and cột$_j(M_P) = \top$.

Như vậy, vector v_1 biểu diễn mô hình tối tiểu $\{r, s\}$ của P.

Nghiên cứu [1] cũng giới thiệu việc tính điểm cố định của mô hình tối tiểu. Khác với nghiên cứu hiện nay, [1] cho một tập rỗng làm vector khởi tạo và tính điểm cố định.chúng ta bắt đầu bằng cách cho vector khởi tạo biểu diễn sự kiện, thay vì biểu diễn sự kiện rõ ràng trên ma trận. Điều này có tác dụng giảm các phần tử khác 0 trong các ma trận khi tính toán điểm cố định. [1] cho phép tồn tại ràng buộc "$\leftarrow q$" trong chương trình trong khí các nghiên cứu hiện tại thì không. Các rang buộc được xử lý như một luật "$p \leftarrow q, \neg p$" trong phân 3.

3.1.2. Xây dựng chương trình thoả điều kiện đơn

Khi một chương trình xác định P chứa 2 luật: r_1: $h \leftarrow b_1 \wedge \ldots \wedge b_m$ and r_2: $h \leftarrow c_1 \wedge \ldots \wedge c_n$, P được biến đổi thành chương trình đơn Q sao cho:

$$Q = (P \setminus \{r_1, r_2\}) \cup \{r_1', r_2', d_1\}$$

Khi đó r_1': $h_1 \leftarrow b_1 \wedge \ldots \wedge b_m$, r_2': $h_2 \leftarrow c_1 \wedge \ldots \wedge c_n$ và d_1: $h \leftarrow h_1 \vee h_2$.

Tóm lại, chương trình khác chương trình đơn được biến đổi thành chương trình đơn như sau.

Định nghĩa 3.1.5: (biến đổi) Cho P là một chương trình logic dạng horn và B_P là cơ sở Herband. Với mỗi $p \in P$, đặt $P_p = \{ r \mid r \in P$ và $head(r) = p\}$ và $R_p = \{ r \mid r \in P_p$ và $\mid P_p \mid = k > 1\}$. Sau đó xác định $S_p = \{p_i \leftarrow body(r) \mid r \in R_p, i = 1,\ldots,k\}$ và $D_p = \{ p \leftarrow p_1 \vee \ldots \vee p_k \}$. Khi đó ta có thể xây dựng chương trình đơn như sau:

$$P' = \underbrace{(P \setminus \bigcup_{p \in B_P} R_p) \cup \bigcup_{p \in B_P} S_p}_{Q} \cup \underbrace{\bigcup_{p \in B_P} D_p}_{D}$$

Như vậy, P' được gọi là một chương trình đơn khi và chỉ khi P' có dạng $P' = Q \cup D$ khi đó Q là một là môt chương trình logic dạng horn thoả điều kiện đơn và D là một tập của d-rules.

Dễ dàng thấy được chương trình đơn P' có môt hình tối tiểu M' sao cho $M = M' \cap B_P$ khi đó M là mô hình tối tiểu của P.

Định nghĩa 3.1.6: (ma trân biểu diễn của chương trình đơn) Cho P' là một chương trình đơn sao cho $P' = Q \cup D$ khi đó Q là một chương trình đơn và D là một tập của luật-d, và $B_{P'} = \{p_1, \ldots, p_{n'}\}$ cơ sở Herband của P'. Thì P' được biểu diễn bằng ma trận $M_{P'} \in \mathbb{R}^{n' \times n'}$ sao cho mỗi phần tử a_{ij} ($1 \leq i, j \leq n'$) trong $M_{P'}$:

1. $a_{ij_k} = 1$ ($1 \leq k \leq m$; $1 \leq i, j_k \leq n'$) nếu $p_i \leftarrow p_{j_1} \vee \ldots \vee p_{j_m} \in D$
2. Trường hợp khác, mỗi luật trong Q được thể hiện trong định nghĩa. 2.4.

Định lý 3.1.2: Cho P' là một chương trình đơn va $M_{P'} \in \mathbb{R}^{n' \times n'}$ là ma trân biểu diễn. Vậy $m \in \mathbb{R}^{n'}$ là vector biểu diễn mô hình tối tiểu của P' khi và chỉ khi $m = FP(M_{P'} v_o)$ khi đó v_o là ma trận khởi tạo của P'.

3.1.3. Thuật toán tìm mô hình tối tiểu

Dựa trên định lý 3.1.2, chúng tôi xây dựng một thuật toán để tính mô hình tối tiểu của chương trình xác định P.

Input: chương trình xác định P, và cơ sở Herband $B_P = \{p_1, \ldots, p_n\}$.
Output: vector u biểu diễn mô hình tối tiểu của P.

Thuật toán 3.1.1:

Bước 1: Biến đổi một chương trình xác định P thành chương trình đơn $P' = Q \cup D$ *với* $B_{P'} = \{p_1, \ldots, p_n, p_{n+1}, \ldots, p_{n'}\}$, khi đó Q là một chương trình đơn và D là một tập luật-d. **Bước 2:** - khởi tạo ma trận $M_{P'} = (a_{ij})_{1 \leq i,j \leq n'}$ biểu diễn cho chương trình đơn P'.	**Bước 3:** *tính toán mô hình tới tiểu của P'* $v := v_o$ $u := \theta(M_{P'} v)$ while $u \neq v$ do $v := u;$

- *K*hởi tạo vector khởi tạo v_o = ($v_{1}, \ldots v_{n'}$) của *P'*.	$u := \theta(M_{P'} v)$ end do; #while return $u[1\ldots n]$;

Ví dụ 3.1.2: Xét $P = \{p \leftarrow q,\ p \leftarrow r \wedge s,\ r \leftarrow s,\ s \leftarrow\ \}$ và $B_P = \{p, q, r, s\}$. Sau đó *P* được biến đổi thành chương trình đơn $P' = Q \cup D$ với $B_{P'} = \{p, q, r, s, t, u\}$:

$Q = \{\ t \leftarrow q,\ \ u \leftarrow r \wedge s,\ \ r \leftarrow s,\ \ s \leftarrow\ \}$

$D = \{\ p \leftarrow t \vee u\ \}$.

$$M_{P'} = \begin{array}{c} \begin{array}{cccccc} p & q & r & s & t & u \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & S & S & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ p \\ q \\ r \\ s \\ t \\ u \end{array}$$

chúng ta có ma trận vuông $M_{P'} \in \mathbb{R}^{6\times6}$ biểu diễn *P'*. Cho $v_0 = (0\ 0\ 0\ 1\ 0\ 0)^T$ là vector biểu diễn sự kiện trong *P'*. Vậy, $v_1 = \theta(M_{P'} v_0) = (0\ 0\ 1\ 1\ 0\ 0)^T$, $v_2 = \theta(M_{P'} v_1) = (0\ 0\ 1\ 1\ 0\ 1)^T$, $v_3 = \theta(M_{P'} v_2) = (1\ 0\ 1\ 1\ 0\ 1)^T$, $v_4 = \theta(M_{P'} v_3) = (1\ 0\ 1\ 1\ 0\ 1)^T = v_3$. Như vậy, v_3 là vector biểu diễn mô hình tối tiểu của *P'*, và $v_3[1\ldots4]$ là vector biểu diễn mô hình tối tiểu của *P*, $\{p, r, s\}$.

Trong thuật toán 2.1, độ phức tạp của $M_{P'} v$ là $\mathbf{O}(n'^2)$. Số lần để lặp lớn nhất của $M_{P'} v$ là $(\boldsymbol{n} + \mathbf{1})$ lần. Vì vậy,độ phức tạp của bước 3 là $\mathbf{O}((n + 1)\times n'^2)$ trong trường hợp xấu nhất.

3.1.4. Tính bằng ma trận con

Trong mục này, dựa trên việc sử dụng điều kiện đơn của một chương trình logic dạng horn, chúng tôi đề xuất ra một phương pháp để giảm độ phức tạp của việc tính toán $u = \theta(M_{P'} v)$.

Định nghĩa 3.1.8: (ma trận con biểu diễn của chương trình đơn) Cho *P'* là một chương trình đơn sao cho $P' = Q \cup D$ khi đó *Q* là một chương trình đơn và *D* là tập hợp của các luật-d, và $B_{P'} = \{p_1, \ldots, p_{n'}\}$ cơ sở Herband của *P'*. Sau đó *P'* được biểu diễn bằng ma trận $N_{P'} \in \mathbb{R}^{n' \times n}$ sao cho mỗi phần tử b_{ij} $(1 \le i \le n', 1 \le j \le n)$ trong $N_{P'}$ tương đương với các phần tử a_{ij} $(1 \le i, j \le n')$ tương ứng trong $M_{P'}$ của định nghĩa.3.1.7. $N_{P'}$ được gọi là *ma trận con* của *P'*.

Lưu ý rằng kích thước $M_{P'} \in \mathbb{R}^{n' \times n'}$ của định nghĩa.3.1.6 bị giảm còn $N_{P'} \in \mathbb{R}^{n' \times n}$ trong định nghĩa.3.1.8.

Định lý 3.1.3: Cho *P* là một chương trình xác định với $B_P = \{p_1, \ldots, p_n\}$, và *P'* một biến đổi chương trình đơn với $B_{P'} = \{p_1, \ldots, p_n, p_{n+1}, \ldots, p_{n'}\}$. Cho $N_{P'} \in \mathbb{R}^{n' \times n}$ là ma trận con của *P'*. cho vector $v \in \mathbb{R}^n$ biểu diễn mô hình *I* của *P*, cho $u = \theta_D(N_{P'} v) \in \mathbb{R}^{n'}$.

Vậy *u* là vector biểu diễn mô hình *J* của *P'* sao cho $J \cap B_P = T_P(I)$.

<u>*Chứng minh:*</u>

Cho $v = (v_1, \ldots, v_n)^T$ khi đó $M^1.v = (x_1, \ldots, x_n, \ldots, x_{n'})^T$ với $x_k = a_{k1}v_1 + \ldots + a_{kn}v_n$ $(1 \le k \le n')$

+ Giả sử $w = \theta(M^1.v) = (w_1, \ldots, w_{n'})^T$

và $u = \theta_D(M^1.v) = \theta_D(w) = (u_1, \ldots, u_{n'})^T$

- Chứng minh: $J \cap B_P \subseteq T_P(I)$

Cho $u_k = 1$ $(1 \le k \le n)$ và $p_k = \text{hàng}_k(u)$. Ta chứng minh: $p_k \in T_p(I)$

(1) <u>Trường hợp 1:</u> Nếu $w_k = u_k = 1$: Vì $w_k = 1$, nên $x_k = a_{k1}v_1 + \ldots + a_{kn}v_n \ge 1$

Cho $\{b_1, \ldots, b_m\} \subseteq \{a_{k1}, \ldots, a_{kn}\}$ với $b_i \ne 0$ $(1 \le m \le n)$. Ta có:

$b_i = 1/m$ $(1 \le i \le m)$ và $b_1 v_{j1} + \ldots + b_m v_{jm} = 1$

Khi đó, tồn tại một luật: $p_k \leftarrow p_{k1} \wedge \ldots \wedge p_{km}$ thuộc P mà $p_{ki} = \text{cột}_j(M^1)$ khi $b_i = a_{kj}$ $(1 \le i \le m)$ và $\{p_{k1}, \ldots, p_{km}\} \subseteq I$ tức là $p_k \in T_P(I)$

Ta chứng minh được $p_k \in T_P(I)$.

(2) <u>Trường hợp 2:</u> nếu $u_k \ne w_k$ thì $w_k = 0$

Vì $w_k = 0$, theo định nghĩa of θ_D :

$\exists\ k_o,\ n + 1 \le k_o \le n',\ w_{ko} = 1$

và $\exists d_{ko} \in$ D, hàng$_{ko}(w) = p_{ko} \in$ body(d_{ko}) và head$(d_{ko}) = p_k$

. d_{ko} có dạng:

$p_k \leftarrow p_{ko1} \vee \ldots \vee p_{koq}$ với $p_{ko} \in \{p_{ko1}, \ldots, p_{koq}\} \subseteq B_{P'} \backslash B_P$

. $w_{ko} = 1$ $(n \le k_o \le n')$: $x_{ko} = a_{ko1}v_1 + \ldots + a_{kon}v_n \ge 1$

Cho $\{b_1, \ldots, b_m\} \subseteq \{a_{ko1}, \ldots, a_{kon}\}$ với $b_i \ne 0$ $(1 \le m \le n)$. Ta có:

$b_i = 1/m$ $(1 \le i \le m)$ và $b_1 v_{j1} + \ldots + b_m v_{jm} = 1$

Khi đó tồn tại một luật: $p_{ko} \leftarrow p_{k1} \wedge \ldots \wedge p_{km}$ thuộc Q

mà $p_{ki} =$ cột$_j(M^1)$ khi $b_i = a_{kj}$ $(1 \le i \le m)$ và $\{p_{k1}, \ldots, p_{km}\} \subseteq I$

Bằng cách chuyển đổi chương trình xác định P thành chương trình dương P', ta có:

$p_k \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \in$ P và $\{p_{k1}, \ldots, p_{km}\} \subseteq$ I

Do đó, $p_k \in T_P(I)$

Ở cả 2 trường hợp 1 và trường hợp 2, ta có:

$p_k =$ hàng$_k(u) \in T_P(I)$ nếu $u_k = 1$ $(1 \le k \le n)$

Khi đó $J \cap B_P \subseteq T_P(I)$

- Chứng minh: $T_P(I) \subseteq J \cap B_P$

Ta chứng minh: nếu $p_k \in T_P(I)$ thì $u_k = 1$ $(1 \le k \le n)$

$p_k \in T_P(I)$, vì vậy $\exists \{p_{k1}, \ldots, p_{km}\} \subseteq I$ và $p_k \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \in$ P $(1 \le m \le n)$

$\{p_{k1}, \ldots, p_{km}\} \subseteq I$, vì vậy $v_{kj} = 1$ $(1 \le j \le m)$

+ Nếu $p_k \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \in$ Q thì:

$p_{ki} \in B_P \Rightarrow \exists\ j,\ p_{ki} =$ cột$_j(M^1)$ $(1 \le i \le m)$ và $a_{kj} = 1/m$ $(1 \le j \le n)$

Do đó, $x_k = a_{k1}v_1 + \ldots + a_{kn}v_n = 1 \Rightarrow w_k = 1 = u_k$

+ Nếu $p_k \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \notin$ Q thì:

$\exists k_o,\ n + 1 \le k_o \le n',\ p_{ko} \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \in$ Q

và $p_k \leftarrow p_{ko1} \vee \ldots \vee p_{koq} \in$ D với $p_{ko} \in \{p_{ko1}, \ldots, p_{koq}\}$

. $p_{ko} \leftarrow p_{k1} \wedge \ldots \wedge p_{km} \in$ Q thì:

$p_{ki} \in B_P \Rightarrow \exists\, j$, p_{ki} = cột$_j(M^1)$ $(1 \le i \le m)$: $a_{koj} = 1/m$ $(1 \le j \le n)$

Vì vậy, $x_{ko} = a_{ko1}v_1 + ... + a_{kon}v_n = 1 \Rightarrow w_{ko} = 1$

Bởi vì $p_k \leftarrow p_{ko1} \vee \ldots \vee p_{koq} \in$ D và hàng$_{ko}(w) = p_{ko} \in \{p_{ko1}, \ldots, p_{koq}\}$

Nên $u_k = 1$ (theo định nghĩa θ_D)

Ta có: nếu $p_k \in T_P(I)$ thì $u_k = 1$ $(1 \le k \le n) \Rightarrow T_P(I) \subseteq J \cap B_P$

Do đó: $J \cap B_P = T_P(I)$.

Cho ma trận $M^1 \in \mathbb{R}^{n' \times n}$, định nghĩa $v_1 = \theta_D(M^1.v_0[1\ldots n])$

$$v_2 = \theta_D(M^1.v_1[1\ldots n])$$

$$v_{k+1} = \theta_D(M^1.\, v_k[1\ldots n])$$

Cho một ma trận $N_{P'} \in \mathbb{R}^{n' \times n}$ và vector khởi tạo v_o của P', xác định $v_1 = \theta_D(N_{P'}v_0[1\ldots n])$ và $v_{k+1} = \theta_D(N_{P'}\, v_k[1\ldots n])$ $(k \ge 1)$. Khi đó, $\exists\, v_{k+1} = v_k$ với $k \ge 1$. Khi $v_{k+1} = v_k$, chúng ta viết $v_k = FP(N_{P'}v_0[1\ldots n])$. Ta thấy được $FP(N_{P'}v_0[1\ldots n])$ biểu diễn mô hình tối tiểu của P.

Tóm lại, cho một chương trình đơn P', giá trị k của $v_k = FP(N_{P'}v_0[1\ldots n])$ không lớn hơn giá trị k của $v_k = FP(M_P v_o)$ trong phần 3.1.2.

Bằng định lý 3.1.3, chúng ta có thể thay thế việc tính toán $u = \theta(M_{P'}v)$ ở bước 3 của thuật toán 3.1.1 bằng $u = \theta_D(N_{P'}v[1\ldots n])$. Độ phức tạp của $N_{P'}v[1\ldots n]$ là $\mathbf{O}(\boldsymbol{n'} \times \boldsymbol{n})$, nó còn nhỏ hơn nhiều $\mathbf{O}(\boldsymbol{n'}^2)$ khi $n << n'$.

Ví dụ 3.1.3: Với chương trình đơn P' của ví dụ 3.1.2, chúng ta có ma trận con $N_{P'} \in \mathbb{R}^{6\times 4}$ biểu diễn P'. Cho ma trận khởi tạo $v_0 = (0\ 0\ 0\ 1\ 0\ 0)^T$ của P', nó trở thành $v_1 = \theta_D(N_{P'}v_0[1\ldots 4]) = (0\ 0\ 1\ 1\ 0\ 0)^T$, $v_2 = \theta_D(N_{P'}v_1[1\ldots 4]) = (1\ 0\ 1\ 1\ 0\ 1)^T$, $v_3 = \theta_D(N_{P'}v_2[1\ldots 4]) = (1\ 0\ 1\ 1\ 0\ 1)^T = v_2$.

$$N_{P'} = \begin{array}{cc} \begin{matrix} p & q & r & s \end{matrix} & \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & S & S \end{pmatrix} & \begin{matrix} p \\ q \\ r \\ s \\ t \\ u \end{matrix} \end{array}$$

Vậy v_2 là vector biểu diễn mô hình tối tiểu của P', và $v_2[1\ldots 4]$ là vector biểu diễn mô hình tối tiểu { p, r, s } của P.

3.2. Chương trình dạng chuẩn (Normal Program)

Chương trình dạng chuẩn được chuyển đổi thành chương trình dạng tuyển và sau đó giải mã chúng ở dạng ma trận cách sử dụng tensor bậc ba. Trong phần này, đầu tiên ta sẽ chuyển đổi chương trình dạng chuẩn thành chương trình dạng Horn và sau đó giải mã chúng ở dạng ma trận như trong Mục 3.1.

3.2.1. Tính toán mô hình ổn định

Một *chương trình dạng chuẩn* P là một tập hữu hạn các *luật* có dạng như sau:

$$h \leftarrow b_1 \wedge \ldots \wedge b_k \wedge \neg b_{k+1} \wedge \ldots \wedge \neg b_m \quad (m \geq 0) \tag{3}$$

Với h và b_j là biến mệnh đề. P được chuyển thành một chương trình dạng Horn bằng cách viết luật ở bên trên lại thành như sau:

$$h \leftarrow b_1 \wedge \ldots \wedge b_k \wedge \overline{b_{k+1}} \wedge \ldots \wedge \overline{b_m} \quad (m \geq 0) \quad . \tag{4}$$

Với $\overline{b_i}$ là một mệnh đề mới tương ứng với b_i. Ta có thể gọi b_i là một mệnh đề và $\overline{b_j}$ là một mệnh đề phủ định

Cho một chương trình dạng chuẩn P và một tập $I \subseteq B_P$, chương trình dạng Horn sau khi biến đổi được ký hiệu là P^+, được gọi là dạng *khẳng định*. Như trong định nghĩa 3.1.6, ta có thể biến đổi P^+ thành một chương trình đơn P'.

Định nghĩa: $\overline{I} = \{\overline{p} \mid p \in B_P \setminus I\}$ và $I^+ = I \cup \overline{I}$.

Định lý 3.2.1 [4]: Cho P là một chương trình dạng chuẩn. Khi đó, I là một mô hình ổn định của P khi và chỉ khi I^+ là mô hình tối tiểu của $P^+ \cup \overline{I}$.

Định nghĩa 3.2.1: (ma trận cho một chương trình dạng chuẩn) Cho P là một chương trình dạng chuẩn với $B_P = \{p_1, \ldots, p_k\}$, và P^+ là dạng khẳng định của nó với $B_{P^+} = \{p_1, \ldots, p_k, \overline{q_{m+1}}, \ldots, \overline{q_n}\}$. Cho P' là một chương trình đơn có được từ P^+ với $B_{P'} = \{p_1, \ldots, p_k, p_{k+1}, \ldots, p_m, \overline{q_{m+1}}, \ldots, \overline{q_n}\}$. Ta có $\{q_{m+1}, \ldots, q_n\} \subseteq \{p_1, \ldots, p_k\}$, và P' có m mệnh đề phủ định và (*n*-*m*) mệnh đề khẳng định. Khi đó, *P'* được biểu diễn bằng ma trận $M_{P'} \in \mathbb{R}^{n\times n}$ sao cho mọi phần tử a_{ij} ($1 \le i, j \le n$) trong $M_{P'}$:

1. $a_{ii} = 1$ với $m + 1 \le i \le n$
2. $a_{ij} = 0$ với $m + 1 \le i \le n$ và $1 \le j \le n$ với mọi $i \ne j$
3. Ngược lại, a_{ij} ($1 \le i \le m$; $1 \le j \le n$) có giá trị như trong định nghĩa 3.1.6.

Theo định nghĩa, mệnh đề phủ định được sinh ra trong $M_{P'}$ tương tự như sự kiện (fact). Ta có thể thấy $a_{ii} = 1$ cho $\overline{q_i}$ biểu diễn luật $\overline{q_i} \leftarrow \overline{q_i}$, nó có nghĩa đó là một "phỏng đoán" cho $\overline{q_i}$.

Định nghĩa 3.2.2: (ma trận khởi tạo) Cho *P* là một chương trình dạng chuẩn và $B_P = \{p_1, \ldots, p_k\}$, *P'* là biến đổi theo chương trình đơn của P (thông qua P^+) và $B_{P'} = \{p_1, \ldots, p_k, p_{k+1}, \ldots, p_m, \overline{q_{m+1}}, \ldots, \overline{q_n}\}$. *Ma trận khởi tạo* $M_o \in \mathbb{R}^{n\times h}$ ($1 \le h \le 2^{n-m}$) được định nghĩa như sau:

- Mỗi hàng của M_o tương ứng với mỗi phần tử của $B_{P'}$,với hàng$_i(M_o) = p_i$ khi $1 \le i \le m$ và hàng$_i(M_o) = \overline{q_i}$ khi $m + 1 \le i \le n$.
- $a_{ij} = 1$ ($1 \le i \le m$, $1 \le j \le h$) khi và chỉ khi sự kiện $p_i \leftarrow$ thuộc *P*; ngược lại, $a_{ij} = 0$.
- $a_{ij} = 0$ ($m + 1 \le i \le n$, $1 \le j \le h$) khi và chỉ khi sự kiện $q_i \leftarrow$ thuộc *P*; ngược lại, a_{ij} có thể bằng 0 hoặc 1.

Cho *P* là một chương trình dạng chuẩn và *P'* là biến đổi theo chương trình đơn của P. Với mà trận chương trình $M_{P'} \in \mathbb{R}^{n\times n}$ và ma trận khởi tạo $M_o \in \mathbb{R}^{n\times h}$. Định nghĩa:

$M_1 = \theta(M_{P'}M_o)$ và $M_{k+1} = \theta(M_{P'}M_k)$ ($k \ge 1$).

Khi đó ,ta sẽ tìm được $M_{k+1} = M_k$ với một giá trị $k \ge 1$ nào đó. Khi $M_{k+1} = M_k$, ta viết $M_k = FP(M_{P'}M_o)$.

Giả sử $M_k = FP(M_{P'}M_o)$ ($k \ge 1$). Cho $u = (a_1 \ldots a_m, a_{m+1} \ldots a_n)^T$ là một vector cột của M_k sao cho $a_j = 1$ (tương ứng với $a_j = 0$) ($m + 1 \le j \le n$) khi và chỉ khi $a_i = 0$

(tương ứng với $a_i = 1$) với $1 \le i \le m$, hàng$_j(M_o) = \overline{q_j}$ và hàng$_i(M_o) = p_i = q_j$. Khi đó, ta có kết quả tiếp tiếp theo.

Định lý 3.2.2: u là vector cột biễu diễn tập I của P' khi và chỉ khi $I \cap B_P$ là mô hình ổn định của P.

3.2.2. Thuật toán để tính toán mô hình ổn định

Dữa trên định lý 3.2, ta có thuật toán tìm mô hình ổn định của một chương trình dạng chuẩn P như sau:

Input: chương trình dạng chuẩn P and tập cơ sở Herband $B_P = \{p_1, \ldots, p_k\}$
Output: Tập vector biểu diễn mô hình ổn định của P.

Thuật toán 3.2.1:

<table>
<tr>
<td>
Bước 1: Biến đổi chương trình dạng chuẩn P thành chương trình đơn P' (thông qua P^+) với $B_{P'} = \{p_1, \ldots, p_k, p_{k+1}, \ldots, p_m, \overline{q_{m+1}}, \ldots, \overline{q_n}\}$

Bước 2:
- Tạo ma trận $M_{P'} \in \mathbb{R}^{n \times n}$ biểu diễn chương trình đơn P'.
- Tạo ma trận khởi tạo $M_o \in \mathbb{R}^{n \times h}$ of P'.

Bước 3: Tính điểm cố định FP($M_{P'}M_o$)
$M := M_o$
</td>
<td>
Step 4: Tìm mô hình ổn định của P
result:={ };
for i from 1 to h do
$v := (a_1, \ldots a_m, a_{m+1}, \ldots, a_n)^T$ is i^{th}-column of M
for j from $m + 1$ to n do
$\overline{q_j}$:= row$_j(M)$;
for i from 1 to m do
if row$_i(M) = q_j$ then
if $a_i + a_j \neq 1$ then
break;
end for; #i
if $i \le m$ then break;
end for; #j
</td>
</tr>
</table>

$U := \theta(M_{P'}M)$
while $U \neq M$ do
$M := U;$
$U := \theta(M_{P'}M)$;
end do;
	if $j \le n$ then break;
	else result := result $\cup$ {v};
	end for;
	return result;

Ví dụ 3.2.1: Xét $P = \{ p \leftarrow q \wedge \neg r \wedge s,\ q \leftarrow \neg t \wedge q,\ q \leftarrow s,\ r \leftarrow \neg t,\ s \leftarrow,\ t \leftarrow \}$ với $B_P = \{p, q, r, s, t\}$. Đầu tiên, P được chuyển đổi thành dạng khẳng định P^+ và một chương trình đơn P' như sau:

- $P^+ = \{ p \leftarrow q \wedge \bar{r} \wedge s,\ \ q \leftarrow \bar{t} \wedge q,\ \ q \leftarrow s,\ \ r \leftarrow \bar{t},\ \ s \leftarrow,\ \ t \leftarrow \}$
- $P' = Q \cup D$ với: $Q = \{ p \leftarrow q \wedge \bar{r} \wedge s,\ q_1 \leftarrow \bar{t} \wedge q,\ q_2 \leftarrow s,\ r \leftarrow \bar{t},\ s \leftarrow,\ t \leftarrow \}$

$$D = \{ q \leftarrow q_1 \vee q_2 \}$$

Ta có ma trận $M_{P'} \in \mathbb{R}^{9\times 9}$ biểu diễn chương trình P' và ma trận khởi tạo $M_o \in \mathbb{R}^{9\times 2}$:

$$\begin{array}{cc} & \begin{matrix} p & q & r & s & t & q_1 & q_2 & \bar{r} & \bar{t} \end{matrix} \\ M_{P'} = & \begin{pmatrix} 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & S \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \bar{r} \\ \bar{t} \end{matrix} \end{array} \qquad M_o = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{matrix} p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \bar{r} \\ \bar{t} \end{matrix}$$

Khi đó $M_1 = \theta(M_{P'}M_o)$, $M_2 = \theta(M_{P'}M_1)$, $M_3 = \theta(M_{P'}M_2)$ trở thành:

$$M_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$M_4 = \theta(M_{P'}M_3) = M_3$ chính là điểm cố định.

Trong trường hợp này, vector cột $u = (1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0)^T$ thỏa mãn điều kiện $a_8 = 1$ khi và chỉ khi $a_3 = 0$ với hàng$_8(u) = \bar{r}$ và hàng$_3(u) = r$. Vector u biểu diễn tập $\{p, q, s, t, q_2, \bar{r}\}$ và $\{p, q, s, t, q_2, \bar{r}\} \cap B_P = \{p, q, s, t\}$ là mô hình ổn định của P.

Ở thuật toán 3.1, độ phức tạp của của phép toán $M_{P'}M$ là $\mathbf{O}(n^2 \times h)$. Số lần lặp của phép toán $M_{P'}M$ nhiều nhất là $(n + 1)$. Do đó, đọ phức tạp ở Bước 3 là $\mathbf{O}((n + 1) \times n^2 \times h)$ trong trường hợp xấu nhất.

3.2.3. Tính toán bằng ma trận con

Chúng ta áp dụng kĩ thuật ma trận con ở định nghĩa.3.1.8 vào ma trận biểu diễn cho chương trình dạng chuẩn. Nghĩa là, thay vì xét một ma trận biểu diễn $M_{P'} \in \mathbb{R}^{n\times n}$ của định nghĩa.3.1, chúng ta xét một ma trận con $N_{P'} \in \mathbb{R}^{n\times n'}$ khi $n' = k + n - m$. Chú ý, $n' << n$ nói chung.

Định nghĩa 3.2.3: (**vector khởi tạo**) Cho P là một chương trình dạng chuẩn với $B_P = \{p_1, \ldots, p_k\}$, và P^+ ở dạng khẳng định với $B_{P+} = \{p_1, \ldots, p_k, \overline{q_{k+1}}, \ldots \overline{q_n}\}$ khi $\{q_{k+1}, \ldots, q_n\} \subseteq \{p_1, \ldots, p_k\}$, thì tập các vector khởi tạo của một dạng phủ định P^+ được định nghĩa như sau:

Cho $v_1 \in \mathbb{R}^k$ là vector biểu diễn sự kiện trong P khi hàng$_j(v_1) = p_j$.

Xét $A \subseteq \mathbb{R}^{n-k}$ khi $A = \{(1\ 0\ \ldots\ 0)^T, (1\ 1\ \ldots\ 0)^T, \ldots., (1\ 1\ \ldots\ 1)^T\}$ với card(A) = 2^{n-k}, và $B = \{v \in A \mid \exists\ i\ (1 \le i \le n\text{-}k)$ s.t. $v[i] = 1$ và $\exists\ j\ (1 \le j \le k)$ s.t. $v_1[j] = 1$ và hàng$_j(v_1) = p$ khi và chỉ khi hàng$_i(v) = \overline{p}$ khi v_1 biểu diễn sự kiện trong $P\}$. đặt $v_2 \in \mathbb{R}^{n-k}$ s.t. $v_2 \in A \backslash B$. tập các vector khởi tạo V của P^+ là:

$$V = \left\{ v \in \mathbb{R}^n \mid v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, v_2 \in A \setminus B \right\} \text{ và } h = \text{card}(V)$$

Theo định lý 2.3, nếu $v \in \mathbb{R}^n$ là vector khởi tạo mô hình I của P^+, và $u = \theta_D(N_{P'} v) \in \mathbb{R}^{n'}$, thì u là một vector biểu diễn mô hình J của P' và $J \cap B_{P+} = T_{P+}(I)$.

Vì lý do này, trong bước 3 của thuật toán 3.1, chúng ta có thể thay thế việc tính toán điểm cố định $FP(M_{P'}M_o)$ thành tính toán điểm cố định $FP(N_{P'}v_o[1\ldots n])$ với vector khởi tạo $v_o \in V$. độ phức tạp của việc tính toán $FP(M_{P'}M_o)$ là $\mathbf{O((}\boldsymbol{n}\mathbf{+1)\times}\boldsymbol{n}^{\mathbf{2}}\mathbf{\times}\boldsymbol{h}\mathbf{)}$. Mặt khác, độ phức tạp của $N_{P'}v_o[1\ldots n]$ là $\mathbf{O(}\boldsymbol{n}\mathbf{\times}\boldsymbol{n'}\mathbf{)}$, khi số lần thực hiện phép nhân lớn nhất là $\mathbf{(}\boldsymbol{n}\mathbf{+1)}$ lần, chúng ta có $|V| = h$, vậy độ phức tạp bước 3 của thuật toán này là $\mathbf{O((}\boldsymbol{n}\mathbf{+1)\times}\boldsymbol{n}\mathbf{\times}\boldsymbol{n'}\mathbf{\times}\boldsymbol{h}\mathbf{)}$ với $n' << n$.

Ví dụ 3.2.2: xét một chương trình dạng chuẩn P và một chương trình đơn P' của ví dụ 3.1.

Chúng ta có ma trận con $N_{P'} \in \mathbb{R}^{9\times 7}$ biểu diễn P':

- $v_1 \in \mathbb{R}^5$ biểu diễn sự kiện trong P, $v_1 = (0\ 0\ 0\ 1\ 1)^T$
- $A = \{(0\ \ 0)^T, (1\ \ 0)^T, (0\ \ 1)^T, (1\ \ 1)^T\}$ với $\text{card}(A) = 2^2 = 4$
- $B = \{(0\ \ 1)^T, (1\ \ 1)^T\}$
- $v_2 \in A \setminus B = \{(0\ \ 0)^T, (1\ \ 0)^T\}$
- V = $\{(0\ 0\ 0\ 1\ 1\ 0\ 0)^T, (0\ 0\ 0\ 1\ 1\ 1\ 0)^T\}$ với h =card(V)= 2.

$$N_{P'} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{ccccccc} p & q & r & s & t & \bar{r} & \bar{t} \end{array} \\ \begin{pmatrix} 0 & 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & S & 0 & 0 & 0 & 0 & S \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ p \\ q \\ r \\ s \\ t \\ \bar{r} \\ \bar{t} \\ q_1 \\ q_2 \end{array}$$

Tính điểm cố định $FP(N_{P'}\, v_o)$ ($v_o \in V$):

(i) Cho $v_o = (0\ 0\ 0\ 1\ 1\ 0\ 0)^T$: $v_1 = \theta_D(N_{P'} v_o) = (0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1)^T$

$$v_2 = \theta_D(N_{P'} v_1[1\ldots7]) = (0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1)^T = v_1$$

hàng$_3(v_1) = r$ và hàng$_6(v_1) = \bar{r}$ thì $v_1[3] + v_1[6] = 0$, vậy v_1 không biểu diễn mô hình ổn định của P.

(ii) Cho $v_o = (0\ 0\ 0\ 1\ 1\ 1\ 0)^T$: $v_1 = \theta_D(N_{P'} v_o) = (0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T$, $v_2 = \theta_D(N_{P'} v_1[1\ldots7])$ $= (1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T$, $v_3 = \theta_D(N_{P'} v_1[1\ldots7]) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T = v_2$.

hàng$_3(v_2) = r$ và hàng$_6(v_2) = \bar{r}$ thì $v_2[3] + v_2[6] = 1$

hàng$_5(v_2) = t$ và hàng$_7(v_2) = \bar{t}$ thì $v_2[5] + v_2[6] = 1$

v_2 biểu diễn tập $\{p, q, s, t, \bar{r}\}$ và $\{p, q, s, t, \bar{r}\} \cap B_P = \{p, q, s, t\}$ là mô hình ổn định

Chương 4. KẾT QUẢ THỬ NGHIỆM

Trong chương này, chúng tôi sẽ tiến hành việc thực nghiệm các thuật toán tìm mô hình tối tiểu của một chương trình logic xác định và tập hợp của các mô hình ổn định một chương trình dạng horn và dạng chuẩn bằng phương pháp MD-điều kiện ban đầu, phương pháp SD-điều kiện và phương pháp ma trận con của chúng tôi. So sánh và đưa ra kết luận. Thử nghiệm được tiến hành trên hệ thống có cấu hình như sau:

- Operating system: Window 10.
- CPU: Intel® Core™ i5-8400 <4.0 GHz /14nm / Cores = 6 / Threads = 6 / Cache = 9 MB>, Memory 8GB, DDR-2400
- GPU: GeForce GTX1060 GDDR5 6GB
- Implementation language: Maple 2018, 64 bit [8].

4.1. So sánh trên chương trình logic dạng horn

Trong thử nghiệm này, cho giá trị $n = |B_P|$ là kích thước của tập cơ sở Herband B_P và giá trị $m = |P|$ là số lượng các luật trong P, các luật được tạo ngẫu nhiên như trong Bảng 1:

Bảng 1.Tỉ lệ của các luật trong *p* theo số lượng của các biến mệnh đề trong phần thân

Số lượng các phần tử trong body	0	1	2	3	4	5	6	7	8
Số lượng các luật (theo tỉ lệ %)	x, tại x < n/3	4%	4%	10%	40%	35%	4%	2%	~1%

Trong mục này, chúng tôi sẽ tiến hành việc thực nghiệm và so sánh các thuật toán tìm mô hình tối tiểu của một chương trình logic xác định và thuật toán tìm các mô hình tối thiểu của một chương trình logic dạng tuyển bằng phương pháp các gốc dựa trên việc tính toán ma trận tensor, T_P-operator, chương trình đơn và tính toán ma trận con.

Dựa vào cặp giá trị (*n*, *m*), một chương trình xác định *P* sẽ được tạo ra một cách ngẫu nhiên. Sử dụng Bổ đề 2.2.1, ta chuyển đổi *P* thành một chương trình xác định *P'* thỏa mãn MD-điều kiện. Từ chương trình xác định P ban đầu, ta chuyển đổi P thành một chương trình đơn $P' = Q \cup D$ với *Q* là một chương trình đơn và *D* là một tập d-rules.

Bảng 1 là kết quả của việc thử nghiệm thuật toán 2.2.1, 3.1.1 và 3.1.4 trên *P'* để tìm mô hình tối tiểu của một chương trình *P*. Có 2 bước quan trọng trong thuật toán 2.2.1, 3.1.1 và 3.1.4: Bước 1 để tạo ma trận $M_{P'}$ biểu diễn chương trình *P'*, và bước 3 để tìm điểm cố định. Ta so sánh 4 phương pháp:

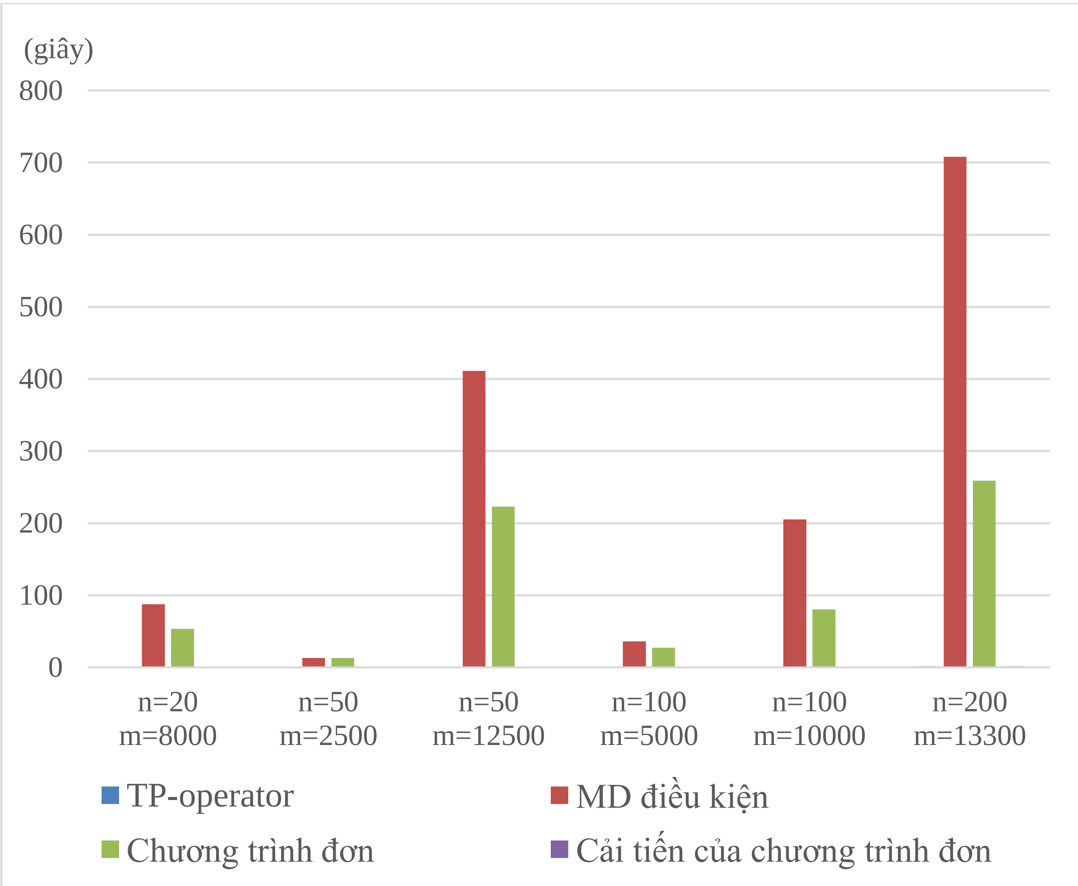
– **Phương pháp 1**: Sử dụng T_P-operator trên chương trình *P*.
– **Phương pháp 2**: tính $\theta(M_{P'}v)$ bằng thuật toán 2.2.1.
– **Phương pháp 3:** tính toán đơn giản một chương trình *P'* bằng thuật toán 3.1.1.
– **Phương pháp 4:** sử dụng tính toán ma trận con trên một chương trình *P'* như ở Mục 3.1.4. Biểu đồ 1 so sánh thời gian tính toán điểm cố định trên chương trình xác định.

Bảng 2. Kết quả thử nghiệm trên chương trình dạng Horn

Data	n	m	**T_P-operator**	**MD điều kiện**	**Chương trình đơn**	**Cải tiến của chương trình đơn**
1	20	400	0.016	0.266	0.344	0.015

2	20	8000	0.125	87.437	53.422	0.141
3	50	1250	0.062	3.594	2.812	0.062
4	50	2500	0.11	13.156	12.766	0.156
5	50	12500	0.391	411.234	223.125	0.406
6	100	5000	0.281	35.844	27.125	0.343
7	100	10000	0.672	205.047	80.391	0.625
8	200	400	0	0.141	0.047	0.016
9	200	13300	1.359	708.125	258.891	1.547

Kết quả trên là thời gian (giây) tính toán điểm cố định.



Biểu đồ 1. Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình xác định

Theo kết quả thực nghiệm trên, ta có thể thấy: Phương pháp sử dụng SD-điều kiện hiệu qua hơn gần gấp 2 lần so với phương pháp MD-điều kiện và càng nhanh hơn khi dữ liệu càng lớn. Phương pháp Sử dụng T_P-operator.

4.2. So sánh trên chương trình logic dạng chuẩn

Trong thử nghiệm này, chương trình dạng tuyển *P* được tạo ngẫu nhiên như sau:

- Cho giá trị $n = |B_P|$ là kích thước của tập cơ sở Herband B_P.
- Cho giá trị $m = |P|$ là số lượng các luật trong *P*.

Về phần đầu (vế trái) của các luật:

Bảng 3. Tỉ lệ các luật trong *P* dựa trên số lượng các biến mệnh đề trong vế trái (Head)

Số lượng phần tử trong head	3	2	1
Số lượng luật	x với x ≤ 3	y với y ≤ 9	còn lại

Do đó, ta có số chương trình dạng Horn có thể được tách từ *P* là
$k \leq 2^9.3^3 = 13824$.

Về vế phải (body) của luật: Tỷ lệ số lượng các luật dựa trên số lượng của các biến mệnh đề trong vế phải như Bảng 1.

Do đó, chương trình *P* có nhiều hơn 95% số luật chứa nhiều hơn một biến mệnh đề trong phần thân.

Dựa trên cặp giá trị (*n*, *m*), một chương trình dạng tuyển *P* sẽ được tạo ra một cách ngẫu nhiên. Ta chuyển đổi *P* thành chương trình *P'* thỏa mãn MD-điều kiện MD. Gọi *k* là số lượng của chương trình dạng Horn SP_h được tách từ *P'*. Từ P ban đầu ta chuyển thành một chương trình dạng chuẩn. Sau đó, ta chuyển đổi *P* thành một chương trình đơn *P'* với *Neg* âm trông chương trình *P*.

Bảng 4 là kết quả của việc thử nghiệm thuật toán 2.3.1, 3.2.1 và 3.2.3 trên *P'* để tìm tập mô hình tối thiểu. Có 3 bước quan trọng trong thuật toán 2.3.1: Bước 2 để tạo một tensor $U^{P'}$ biểu diễn một chương trình dạng tuyển *P'*, bước 3.3 để tìm ma trận biểu diễn mô hình tối tiểu của các chương trình dạng Horn SP_h, và bước 4 để tìm tập các vector biểu diễn mô hình tối thiểu của *P'*. Tương tự, Có 3 bước quan trọng ở Thuật toán 3.2.1: Bước 2 để tạo một ma trận biểu diễn biểu diễn một chương trình đơn *P'*, Bước 3 để tính toán điểm cố định và Bước 4 để tìm tập các vector dùng để biểu diễn mô hình ổn định. Ta so sánh 4 phương pháp sau:

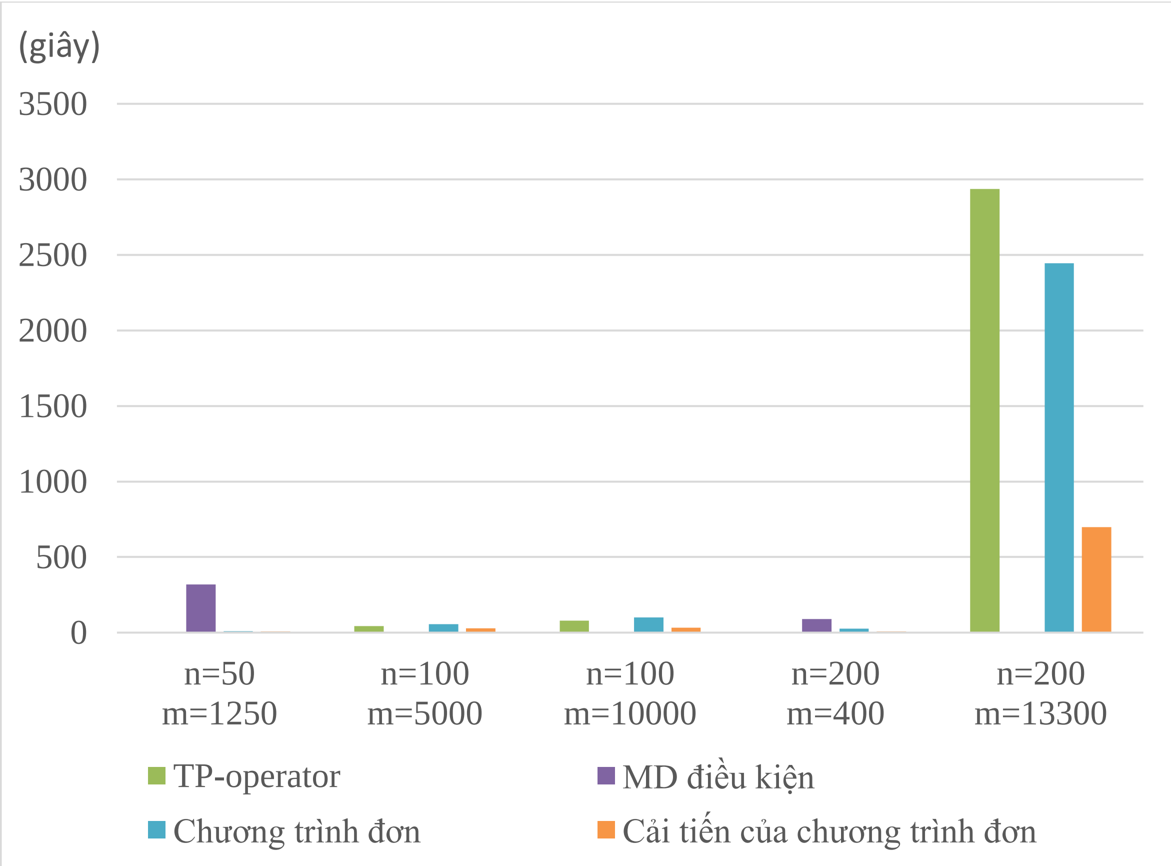
- **Phương pháp 1**: Tính toán bằng T_P-operator (sử dụng Định lý 3.2.1).
- **Phương pháp 2**: Phương pháp tính $\theta(U^{P'}M)$ bằng thuật troán 2.3.1 cho bước 3.3 của chương trình dạng tuyển.
- **Phương pháp 3:** tính toán đơn giản trên một chương trình *P'* bằng Thuật toán 3.2.1.
- **Phương pháp 4:** sử dụng tính toán ma trận con trên chương trình *P'* như ở Mục 3.2.3 của Bước 3 của chương trình dạng chuẩn

Bảng 4. Kết quả thử nghiệm của chương trình dạng chuẩn

Data	n	m	*Neg*	**T_P-operator**	**MD điều kiện**	**Chương trình đơn**	**Cải tiến của chương trình đơn**
1	20	40	9	0.078	0.406	0.172	0.115
2	20	400	6	0.656	36	0.281	1.485
3	50	100	7	0.562	1.156	0.047	0.156
4	50	1250	6	3.797	319.328	9.922	6.172

5	100	200	10	1.61	11.078	5.906	5.484
6	100	5000	7	42.234	>3000	56.766	27.36
7	100	10000	7	78.234	>3000	101.422	32.391
8	200	400	10	2.797	90.11	26.5	6.172
9	200	13300	10	2937.125	>3000	2445	697.156

Kết quả trên là thời gian (giây) tính toán điểm cố định. Với *Neg* số phần tử phủ định trong chương trình *P*.



Biểu đồ 2. Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình dạng chuẩn

Phương pháp 2 tính $\theta(U^P M)$ bằng thuật toán 2.3.1 cho kết quả chậm nhất và chậm hơn các phương pháp còn lại ở tất cả các trường hợp. Phương pháp 1 tính toán bằng T_P-operator cho kết quả tốt nhất ở các dữ liệu nhỏ nhưng đến các dữ liệu lớn cụ thể từ n = 100 và m gấp 2 lần n thì phương pháp ma trận con cho kết quả tốt hơn rất nhiều, đặc biệt với n = 200 và m = 13300 phương pháp ma trận con cho kết quả nhanh gấp 4 lần so với tính toàn bằng T_P-operator **.**

Chương 5. KẾT LUẬN

5.1. Kết quả đạt được

Trong khoá luận này, chúng tôi đề xuất các phương pháp đại diện cho lập trình logic dựa trên đại số đa tuyến tính. Chúng tôi phát triển các thuật toán mới để tính các ngữ nghĩa lập trình logic trong đại số tuyến tính và các phương pháp cải tiến để tăng tốc các thuật toán đó.

Kết quả thử nghiệm cho thấy rằng tính toán ma trận con thì nhanh hơn tính toán mô hình tối tiểu, trong khi tính toán ma trận đơn giản thì thường tốt hơn ma trận con trong việc tính toán mô hình đơn giản. Ta đã biết rằng mô hình tối tiểu của một chương trình xác định thì được tính bằng O(N) [5] với N là kích thước (số lượng các chữ) của một chương trình. Vì việc tính toán ma trận con tiêu tốn O($n^2 \times n'$),nó sẽ có hiệu quả khi $n^2 \times n' < N$, nghĩa là., kích thước chương trình thì lớn với một số lượng phần tử nhỏ. Để tính toán các mô hình ổn định của một chương trình dạng chuẩn, mặc dù kích thước ma trận khởi tạo lớn, trong đó có nhiều phần tử giá trị 0. Chúng tôi có thể cái thiện phương pháp biểu diễn ma trận, điều này cũng mang lại lợi thế lưu trữ. Để tính toán mô hình ổn định, ta cần thêm sự tối ưu hóa và so sánh với các giải pháp hiện hành như clasp [6] và DLV [7].

5.2. Hướng phát triển

Hiệu suất của việc thực hiện đại số tuyến tính của chúng tôi phụ thuộc rất nhiều vào việc vận dụng ma trận. Chúng tôi đã sử dụng Maple để triển khai, nhưng các phương pháp của chúng tôi có thể được thực hiện bằng các ngôn ngữ và kiến trúc máy tính khác. Hiện nay dự kiến sẽ có nhiều nền tảng mạnh hơn được phát triển cho tính toán đại số tuyến tính trong tương lai gần.. Các phương pháp của chúng tôi sẽ được chứng minh là xứng đáng khi những công nghệ này trở nên khả dụng. Tuy nhiên, tính toán đại số tuyến tính cho lập trình logic chỉ vừa mới bắt đầu, còn rất nhiều thứ cần được cải tiến và tối ưu hóa. Bên cạnh đó, chúng tôi cũng nghiên cứu kỹ thuật sử dụng tính toán song song trên GPU để tính toán các chương trình logic.

BÀI BÁO KHOA HỌC CỦA ĐỀ TÀI

Nguyễn Đình Hiển, Trương Ngọc Kha, Huỳnh Đăng Khoa, Trần Anh Dũng, "*Phương pháp cải tiến tính toán chương trình logic theo tiếp cận đại số tuyến tính*", bài báo được chấp nhận đăng trong kỷ yếu của Hội thảo quốc gia lần thứ 21, Một số vấn đề chọn lọc của Công nghệ thông tin và Truyền thông, Trường Đại học Hồng Đức, Thanh Hóa, tháng 7/2018

TÀI LIỆU THAM KHẢO

[1] C. Sakama, K. Inoue, T. Sato, "*Linear Algebraic Characterization of Logic Programs*", Proceeding of 10th International conference on Knowledge Science, Engineering and Management (KSEM 2017), LNAI 10412, pp.530-533, Springer, Melbourne, Australia (2017).

[2] Gerhard Brewka, Thomas Eiter, Mirosław Truszczynski, "*Answer Set Programming at a Glance*", Communications of the ACM, Vol. 54, No. 12 (2011).

[3] M.H. van Emden, R.A. Kowalski, "*The semantics of predicate logic as a programming language*",.Journal of the ACM 23(4), pp. 733–742 (1976).

[4] F. Lin, "*From satisfiability to linear algebra*", Invited talk, 26th Australian Joint Conference on Artificial Intelligence (2013).

[5] William F. Dowling, Jean H. Gallier, "*Linear-time algorithms for testing the satisfiability of propositional Horn formulae''*, Journal of Logic Programming 1(3), 267-284 (1984).

[6] clasp: https://potassco.org/clasp/

[7] DLV system: http://www.dlvsystem.com/dlv/

[8] Maple: https://www.maplesoft.com/support/install/maple2018_install.html

[9] T. Kolda, B. Bader, "*Tensor Decompositions and Applications*", SIAM Review 51(3), pp. 455-500 (2009).

[10] Gelfond, M. and Lifschitz, V. "*The stable model semantics for logic programming*". Logic Programming: The 5th International Conference and Symposium. R.A. Kowalski and K. Bowen, Eds. MIT Press, Cambridge, MA, pp. 1070–1080 (1988).

[11] L. Serafini, A. Garcez, "*Learning and Reasoning with Logic Tensor Networks*", Proceeding of 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016), LNAI 10037, pp. 334-348, Springer, Genoa, Italy (2016).

[12] L. Serafini, I. Donadello, A. Garcez, "*Learning and Reasoning with Logic Tensor Networks: Theory and application to semantic image interpretation*", Proceeding of 32nd ACM SIGAPP Symposium On Applied Computing (SAC 2017), pp. 125-130, Marrakech, Morocco (2017)

[13] B. Yang, W. Yih, X. He, J. Gao, L. Deng, "*Embedding entities and relations for learning and inference in knowledge bases*", Proceeding of Third International Conference Learning Representations (ICLR 2015), San Diego, USA (2015)

[14] V. Saraswat, *Reasoning 2.0 or machine learning and logic–the beginnings of a new computer science*, Data Science Day, Kista Sweden (2016)

[15] T. Sato, *A linear algebraic approach to Datalog evaluation*, Theory and Practice of Logic Programming, Vol. 17, No. 3, pp. 244–265 (2017)

[16] E. Grefenstette, "*Towards a Formal Distributional Semantics: Simulating Logical Calculi with Tensors*", Proceeding of Second Joint Conference on Lexical and Computational Semantics (*SEM), Vol. 1, pp. 1–10, Atlanta, USA (2013).

[17] B. Coecke, M. Sadrzadeh, S. Clarky, "*Mathematical Foundations for a Compositional Distributional Model of Meaning"*, Linguistic Analysis, vol. 36, pp. 345–384 (2011)

[18] C. Bell, A. Nerode, T.Ng. Raymond, V.S. Subrahmanian, "*Mixed integer programming methods for computing nonmonotoic deductive database*", Journal of ACM 41(6), pp. 1178-1215 (1994).