Tạp chí Tin học và Điều khiển học, T. 17, S. 4 (2001), 87–96

PHÂN TÍCH MỘT SỐ PHƯƠNG PHÁP XỬ LÝ VÒNG LẶP VÔ HẠN TRONG QUÁ TRÌNH ƯỚC LƯỢNG CÂU TRUY VẤN ĐỐI VỚI CHƯƠNG TRÌNH DATALOG

LÊ MẠNH THẠNH, TRƯƠNG CÔNG TUẤN

Abstract. Resolution is the main technique used by query answering systems for logic programs. This paper analyses and compares two methods that avoid infinite loops in the process of query evaluation for logic programs with finite models. We present some search strategies and discuss a particular tabulation technique, SLG resolution, and a particular transformation technique, Magic Templates. They are all goal-oriented and can be applied to evaluate queries for the Datalog programs. The primary differences between direct implementations of both approaches are in the maintenance of data structures.

Tóm tắt. Phép phân giải là kỹ thuật chính được các hệ thống trả lời câu truy vấn sử dụng trong các chương trình logic. Bài báo tập trung phân tích và so sánh hai phương pháp nhằm ngăn chặn các vòng lặp vô hạn trong quá trình ước lượng câu truy vấn đối với các chương trình logic với mô hình hữu hạn. Chúng tôi trình bày các chiến lược tìm kiếm, thảo luận về phép biến đổi ma tập và phép ước lượng bảng SLG. Cả hai phương pháp này đều là những thuật toán hướng đích, chúng ta có thể áp dụng để ước lượng câu truy vấn đối với chương trình Datalog. Sự khác nhau cơ bản trong việc thực hiện của cả hai cách tiếp cận này là về mặt cấu trúc dữ liệu.

1. MỞ ĐẦU

Các kỹ thuật để trả lời câu truy vấn đối với các chương trình logic đã được nghiên cứu nhiều trong các năm qua và có thể tìm thấy nhiều công trình nghiên cứu [2, 4 - 8, 11]. Những kỹ thuật này được áp dụng theo hai cách khác nhau, thường được gọi là trên xuống (top-down) và dưới lên (bottom-up). Các phương pháp top-down có vẻ là các phương pháp trực giác do điểm khởi đầu của việc tính toán là từ đích truy vấn, và chúng sẽ không tính các fact không thích hợp với câu truy vấn. Tuy nhiên các kết quả trung gian có thể được tính toán lặp đi lặp lại nhiều lần, chẳng hạn với phương pháp ước lượng SLD [5], phương pháp này là đầy đủ, nghĩa là tất cả câu trả lời đúng được biểu diễn trong cây SLD. Cây SLD đã tạo ra một sự phân chia chính xác trong không gian tìm kiếm: Cần tính toán gì và thự tự tính toán là như thế nào. Một điều đáng tiếc đối với phương pháp này là nó không hiệu quả, việc tính toán trên cây SLD có thể kéo dài vô tận. Các phương pháp bottom-up đảm bảo tính kết thúc trong quá trình tính toán lời giải của câu truy vấn, nhưng điều này không có nghĩa là nó hiệu quả. Chúng thường không định hướng đích, nhiều fact không liên quan đến câu truy vấn cũng được tính toán. Một số phương pháp mở rộng để trả lời câu truy vấn được đưa ra trong thời gian gần đây mà mục đích là đưa ra được một chiến lược tìm kiếm hướng đích như trong SLD, đồng thời có tính hiệu quả là đảm bảo kết thúc quá trình tính toán câu trả lời truy vấn. Điển hình của các phương pháp này là phương pháp biến đổi ma tập và phương pháp ước lượng bảng (xem [2, 6, 11]).

Trong bài báo này chúng tôi tập trung vào một số phương pháp ước lượng câu truy vấn đối với các chương trình với các mô hình hữu hạn, cụ thể chúng tôi thảo luận các phương pháp ước lượng câu truy vấn trên chương trình Datalog: phép phân giải SLD, phép biến đổi ma tập và phép ước lượng bảng SLG, đồng thời cũng nêu lên mối quan hệ giữa các phương pháp.

2. MỘT SỐ ĐỊNH NGHĨA VÀ KHÁI NIỆM CƠ BẢN

Trong phần này giới thiệu một số định nghĩa và khái niệm cơ bản, chi tiết đầy đủ hơn có thể xem trong [10].

Định nghĩa 2.1. Một term được định nghĩa đệ qui như sau:

- Một hằng hoặc biến là một term.
- Nếu f là kí hiệu hàm bậc n và $t_1, t_2, \ldots, t_n$ là các term thì $f(t_1, t_2, \ldots, t_n)$ là một term.

Term cơ sở là term không chứa biến.

Nếu p là kí hiệu vị từ bậc n và $t_1, t_2, \ldots, t_n$ là các term thì $p(t_1, t_2, \ldots, t_n)$ được gọi là một nguyên tử. Một literal là một nguyên tử hoặc phủ định của một nguyên tử.

Định nghĩa 2.2:

- Một *qui tắc* là một công thức logic có dạng:

$$p \leftarrow q_1 \wedge q_2 \wedge \cdots \wedge q_n ,$$

trong đó $p, q_1, \ldots, q_n$ là các nguyên tử. Vị từ p được gọi là đầu, $q_1 \wedge q_2 \wedge \cdots \wedge q_n$ được gọi là thân và các vị từ $q_1, q_2, \ldots, q_n$ được gọi là các đích con của qui tắc. Ý nghĩa của qui tắc này là "nếu $q_1, q_2, \ldots, q_n$ đúng thì p cũng đúng". Nếu thân của qui tắc là rỗng thì nó được gọi là một fact, ta bỏ qua kí hiệu "$\leftarrow$".

Các vị từ được định nghĩa bởi các quy tắc gọi là *vị từ IDB*. Mỗi vị từ IDB ứng với một quan hệ IDB, các bộ của quan hệ IDB không có sẵn và chỉ được tạo ra từ các qui tắc. *Vị từ EDB* là vị từ không được định nghĩa qua các qui tắc mà chính là những quan hệ được lưu trong cơ sở dữ liệu. Tương ứng mỗi vị từ EDB là một quan hệ EDB.

- *Một đích (goal) hay câu truy vấn (query)* có dạng như sau: ?- $q_1 \wedge q_2 \wedge \cdots \wedge q_n$.
- *Quy tắc Datalog* là qui tắc mà các term của nó gồm hằng hoặc biến và không chứa kí hiệu hàm.
- Một chương trình Datalog là một tập hữu hạn các qui tắc Datalog.

3. ƯỚC LƯỢNG SLD (LINEAR SELECTION RESOLUTION FOR DEFINITE CLAUSES)

Chúng tôi giới thiệu phương pháp ước lượng SLD (xem [5]) được thiết kế đặc biệt cho việc tính các qui tắc Datalog theo hướng từ trái sang phải.

Định nghĩa 3.1. (Cây SLD): Cho P là một phương trình Datalog và Q là mộc đích truy vấn. Cây SLD của đích truy vấn Q đối với chương trình P được xác định:

- Mỗi nút của cây là một đích và gắn cùng với nó bởi một phép thay thế.
- Đích Q là nút gốc, phép thay thế của nó là rỗng.
- Gọi đích $Q' = \leftarrow q_1 \wedge q_2 \wedge \cdots \wedge q_n$ $(n \geq 1)$ là một nút trong cây. Lúc đó Q' có một nút con đối với mỗi qui tắc r sao cho đầu của r và q_1 là có thể hợp nhất được. Đặt: $H \leftarrow b_1 \wedge B_2 \wedge \cdots \wedge B_n$ là một biến thể của r bằng cách dùng các biến mới. Nút con Q'' là:

$$\leftarrow (B_1 \wedge B_2 \wedge \cdots \wedge B_n \wedge q_2 \wedge \cdots \wedge q_n)\theta,$$

  trong đó θ là phép hợp nhất tổng quát (mgu) của q_1 và H. Phép thay thế tính toán ở Q'' là $\phi\theta$ trong đó ϕ là phép thay thế tính toán ở Q'.
- Nút kết thúc là nút không có nút con.

Định nghĩa 3.2. Việc ước lượng câu truy vấn trên chương trình Datalog dựa vào cây phân giải SLD được gọi là phép ước lượng SLD.

Phép ước lượng SLD là một chiến lược xử lí câu truy vấn theo kiểu top-down. Quá trình ước lượng bắt đầu từ đích truy vấn và lặp lại các phép thay thế thân của qui tắc đối với một trường hợp riêng của đầu trong đích được chỉ ra. Quá trình ước lượng câu truy vấn sẽ thành công nếu tất cả literal trong đích là được tìm ra, có ý nghĩa khi đích là nút kết thúc.

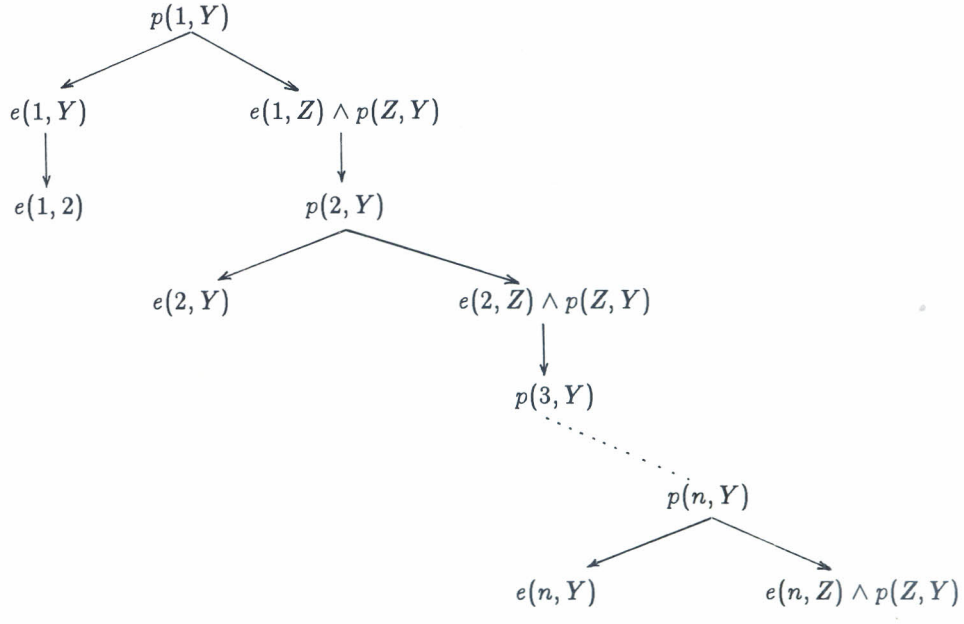
Ví dụ 1. Xét chương trình P sau đây:

$$r_1 : p(X,Y) \leftarrow e(X,Y),$$
$$r_2 : p(X,Z) \leftarrow e(X,Y) \wedge p(Y,Z).$$

Câu truy vấn (Q): ?-$p(1,Y)$.

Quan hệ EDB của vị từ e được cho bởi tập hợp $E = \{(1,2),\ldots,(n-1,n)\}$.

Cây SLD đối với câu truy vấn $p(1,Y)$ như sau:



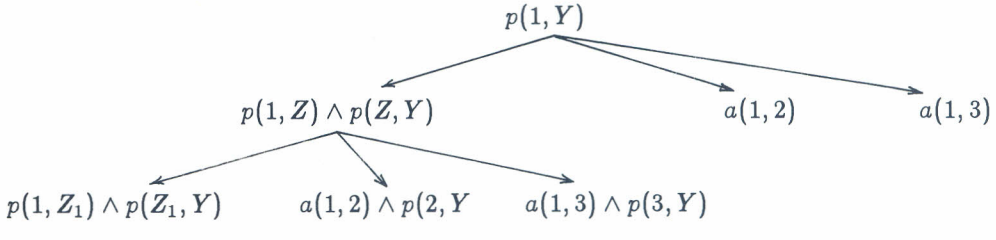
Phép ước lượng SLD có thể sẽ kéo dài vô tận trong trường hợp cây SLD là vô hạn. Chúng ta xét ví dụ sau:

Ví dụ 2. Xét phương trình P sau đây:

$$r_1 : p(X,Y) \leftarrow p(X,Z) \wedge p(Z,Y),$$
$$r_2 : p(X,Y) \leftarrow a(X,Y).$$

Câu truy vấn: ?- $p(1,Y)$.

Trong đó a là vị từ EDB, giả sử quan hệ A của vị từ a gồm các bộ $(1,2)$, $(1,3)$, $(2,3)$. Cây phân giải SLD đối với phương trình này và câu truy vấn đã cho như sau:



Nhánh bên trái nhất của cây phân giải này là kéo dài vô hạn, vì vậy phép ước lượng SLD sẽ đi vào vòng lặp vô tận khi tìm kiếm các lời giải câu truy vấn. Trong các phần tiếp theo, chúng tôi trình bày hai phương pháp ước lượng để tránh gặp phải trường hợp này.

4. PHƯƠNG PHÁP MA TẬP

Các phương pháp ước lượng theo cách tiếp cận bottom-up có ưu điểm là việc tìm lời giải câu truy vấn đảm bảo kết thúc đối với các chương trình Datalog. Tuy nhiên, nó không xem xét câu truy vấn trong suốt quá trình ước lượng, tức là việc tính toán không được gắn liền với câu truy vấn như thường xảy ra trong các phương pháp top-down, từ đó dẫn đến việc tính toán nhiều fact không liên quan đến câu truy vấn cần trả lời. Một số tiếp cận kiểu bottom-up bao gồm định hướng đích (goal-oriented) được đưa ra trong nhiều công trình nghiên cứu, trong đó phương pháp nổi tiếng là phương pháp ma tập (xem [11]). Ý tưởng chính của phương pháp này là mô phỏng sự lan truyền các trị buộc được tạo ra trong ước lượng top-down của câu truy vấn. Sự lan truyền này nhận được bằng cách viết lại chương trình gốc ban đầu. Trong mỗi qui tắc gốc một điều kiện mới được thêm vào để hạn chế sự ứng dụng qui tắc. Các điều kiện này được biết như là các quan hệ lọc. Một tập qui tắc mới được tạo ra để mô phỏng sự lan truyền các trị buộc.

Chúng tôi trình bày một số khái niệm cơ bản trước khi nêu lên thuật toán ma tập.

4.1. Hoa văn

Chúng ta có thể hình dung hoa văn là cách chú thích trên các vị từ để cung cấp thông tin về các vị từ sẽ được sử dụng như thế nào trong quá trình ước lượng câu truy vấn.

Định nghĩa 4.1.1. Một đối tượng của một đích con trong quy tắc r được gọi là buộc nếu trong suốt quá trình ước lượng câu truy vấn, mọi đích được tạo ra từ đích con này có một tập các hằng trong vị trí đối này. Ngược lại, đối được gọi là tự do.

Định nghĩa 4.1.2.

- Một hoa văn (hoặc mẫu buộc) của vị từ $p(t_1, t_2, \ldots, t_k)$ là một chuỗi các kí tự b, f có chiều dài k. Nếu kí hiệu thứ i của hoa văn là b thì đối thứ i của p là buộc, nếu kí tự thứ i của hoa văn là f thì đối thứ i của p là tự do. Chỉ có các vị từ IDB là được hoa văn.
- Cho qui tắc $p \leftarrow q_1 \wedge q_2 \wedge \cdots \wedge q_n$ và w là hoa văn của vị từ p, hoa văn α_i của các literal $q_i(t_{i,1}, \ldots, t_{i,n_i})$ được xác định như sau: Nếu $t_{i,j}$ là một hằng hoặc biến đã xuất hiện trong đích con q_k trước đó ($k < i$) hoặc trong một vị trí buộc của p thì $\alpha_i[j] = b$, ngoài ra thì $\alpha[j] = f$ (với $\alpha_i[j]$ là kí hiệu vị trí thứ j của hoa văn).
- Cho chương trình P, chương trình hoa văn của P, kí hiệu là P^{ad}, gồm các qui tắc hoa văn của mọi qui tắc trong P.
- Hoa văn α của câu truy vấn $p(t_1, \ldots, t_n)$ được xác định bởi: $\alpha[i] = b$ nếu t_i là một hằng và $\alpha[i] = f$ nếu ngược lại.

Ví dụ 3. Chương trình Datalog P trong Ví dụ 1 có thể biến đổi thành chương trình hoa văn P^{ad} như sau:

$$?\text{-}\ p^{\text{bf}}(1, Y),$$
$$ar_1 : p^{\text{bf}}(X, Y) \leftarrow e(X, Y),$$
$$ar_2 : p^{\text{bf}}(X, Z) \leftarrow e(X, Y) \wedge p^{\text{bf}}(Y, Z).$$

Các vị từ được hoa văn p^{bf} và p^{fb} là khác biệt nhau và khác với từ không hoa văn p, mặc dầu các qui tắc định nghĩa p^{bf} và p^{fb} nhận được từ các qui tắc định nghĩa p.

4.2. Truyền thông tin ngang

Một chiến lược truyền thông tin ngang SIPS (Sideway Information Passing Strategy) là một quyết định về *cách thức để lan truyền thông tin trong thân qui tắc khi ước lượng qui tắc*, SIPS chỉ rõ các trị buộc của đầu qui tắc được dùng như thế nào, thứ tự mà các đích con trong thân sẽ được tính và cách thức để các trị buộc này truyền ngang giữa các vị từ của thân quy tắc. Sử dụng SIPS ta có thể truyền các trị buộc của đầu qui tắc và các trị buộc nhận được từ việc ước lượng các đích con trước đó vào các đích con chưa được ước lượng. Điều này tương ứng với việc tính toán các qui tắc trong ước lượng top-down nhưng ở đây quyết định được thực hiện ở thời gian dịch. Nói cách khác, một chiến lược truyền thông tin ngang mô tả cách thức để ước lượng một qui tắc khi tập các đối của đầu là bị buộc vào các hằng. Để thực hiện điều này ta có thể dùng các qui tắc được hoa văn.

Định nghĩa. Một chiến lược truyền thông tin ngang SIPS(r, α) đối với qui tắc r và hoa văn của đầu qui tắc α của r là một đồ thị có hướng được gán nhãn. Các cung có dạng $N \xrightarrow{S} \{q\}$ với N là tập các vị từ trong chương trình, S là tập các biến và q là là một vị từ đơn. Các cung và nhãn chỉ định cách thức thông tin được truyền giữa các đích con. Các cung của đồ thị đưa ra một thứ tự bộ phận mà ở đó các đích con được ước lượng, còn các nhãn chỉ định thông tin được truyền ngang từ đích con này đến đích con khác.

Để ý rằng mỗi chiến lược SIP xác định một mẫu ẩn các đối buộc hoặc tự do với mỗi vị từ được ước lượng. Nhiệm vụ của quá trình hoa văn là làm hiện ra các mẫu ẩn này. Chẳng hạn, ta xem ví dụ sau: Xét một qui tắc r với đầu là p và thân q và một câu truy vấn p^{α}. Từ qui tắc này ta tạo ra một phiên bản hoa văn của thân q như sau: Chắc chắn tồn tại một SIP phù hợp với hoa văn α từ câu truy vấn. Điều này có nghĩa là trong SIP qui tắc r xuất hiện với các đối p buộc hoặc tự do theo hoa văn α. Bấy giờ, mỗi vị từ IDB trong thân qui tắc được thay bởi một phiên bản hoa văn được xác định bởi SIP. Nếu phiên bản này là mới thì ta phải tạo ra phiên bản hoa văn khác đối với các qui tắc định nghĩa nó.

Ta minh họa ví dụ về một chiến lược SIP để truyền ngang các trị buộc như trong Prolog. Xem qui tắc $p(X, Y) \leftarrow s(X, Z) \wedge t(Z, Y)$. Câu truy vấn là $?p(a, X)$, tương đương với truy vấn hoa văn $?p^{\mathrm{bf}}(a)$. Nếu p có hoa văn bf thì s có hoa văn bf bởi nó nhận biến X bị buộc từ đầu quy tắc và t có hóa văn bf vì nó nhận Z từ vị từ s. Lúc đó phiên bản hoa văn của quy tắc sẽ là $p^{\mathrm{bf}}(X, Y) \leftarrow s^{\mathrm{bf}}(X, Z) \wedge t^{\mathrm{bf}}(Z, Y)$. Lúc này nếu không còn qui tắc nào với đích con s có hoa văn bf thì quá trình hoa văn được áp dụng đối với các qui tắc định nghĩa s. Dĩ nhiên, các SIPS khác nhau sẽ xác định các hoa văn khác nhau đối với cùng một qui tắc.

Các hoa văn giải quyết bài toán truyền các trị buộc dư thừa bởi vì vậy bây giờ nó được nhận biết các tham số nào là bị buộc và những tham số nào là tự do.

4.3. Thuật toán ma trận [11]

Định nghĩa 4.3.1.

- Literal magic của literal $p(\bar{t})$ là literal kí hiệu magic_$p(\overline{t^b})$, trong đó $\overline{t^b}$ là kí hiệu các đối bị buộc của p.
- Các qui tắc định nghĩa literal magic_$p(\overline{t^b})$ được gọi là các qui tắc magic.
  Các qui tắc magic chỉ truyền các trị buộc cơ sở.

Thuật toán:

Input: Cho chương trình P, một câu truy vấn q. Gọi P^{ad} là chương trình hoa văn nhận được từ P theo một chiến lược SIP đã cho.

Output: Một chương trình MP^{ad} sao cho khi ước lượng MP^{ad} sẽ cho kết quả câu truy vấn q.

Phương pháp:

1. Đối với mỗi vị từ p với đối là $\bar{t}$ trong chương trình P^{ad}, tạo ra một vị từ mới magic_p với đối $\overline{t^b}$, là các đối bị buộc của vị từ p.
2. Đối với mỗi qui tắc r trong P^{ab} : $p(\bar{t}) \leftarrow q_1(\bar{t}_1) \wedge \cdots \wedge q_n(\bar{t}_n)$ ta sửa đổi thành một qui tắc trong MP^{ad} : $p(\bar{t}) \leftarrow \text{magic\_}p(\overline{t^b}) \wedge q_1(\bar{t}_1) \wedge \cdots \wedge q_n(\bar{t}_n)$.
3. Đối với mỗi qui tắc r trong P^{ad} : $p(\bar{t}) \leftarrow q_1(\bar{t}_1) \wedge \cdots \wedge q_n(\bar{t}_n)$ và với mọi vị từ IDB q_i, ta thêm vào MP^{ab} qui tắc magic:
$$\text{magic\_}q_i(\overline{t_i^b} \leftarrow \text{magic\_}p(\overline{t^b}) \wedge q_1(\bar{t}_1) \wedge \cdots \wedge q_{i-1}(\bar{t}_{i-1}).$$
4. Thêm một fact "hạt nhân" magic_$q(\bar{c})$, trong đó $\bar{c}$ là tập các hằng tương ứng với các đối bị buộc của câu truy vấn.

Tính đúng đắn của thuật toán này được thể hiện bởi định lý sau đây:

4.4. Định lý. *Cho $p(c_1, \ldots, c_n)$ là literal. Lúc đó:*

- *Nếu $MP^{ad} \vdash p(c_1, \ldots, c_n)$ thì $P \vdash p(c_1, \ldots, c_n)$.*
- *Nếu $P \vdash p(c_1, \ldots, c_n)$ và $MP^{ad} \vdash magic[p]$, thì $MP^{ad} \vdash p$.*

4.5. Hệ quả [11] *Cho chương trình P và câu truy vấn q. Dùng thuật toán ma tập để biến đổi chương trình P thành chương trình MP^{ad}, chương trình này sẽ tương đương với P theo nghĩa khi ước lượng MP^{ad} sẽ cho ra cùng kết quả câu truy vấn q.*

4.6. Phương pháp ước lượng ma tập bao gồm hai bước sau đây:

1. Dùng thuật toán ma tập để viết lại chương trình P thành chương trình MP^{ad}.
2. Ước lượng chương trình MP^{ad} bằng các thuật toán kiểu dưới lên như Naive, Seminaive,...

Ví dụ 4. Sử dụng phép biến đổi ma tập ta nhận được chương trình MP^{ad} sau đây:

$$\text{mar}_1 : p(X, Y) \leftarrow \text{mag\_}p^{\text{bf}}(X) \wedge e(X, Y)$$
$$\text{mar}_2 : p(X, Z) \leftarrow \text{mag\_}p^{\text{bf}}(X) \wedge e(X, Y) \wedge p^{\text{bf}}(Y)$$
$$\text{mar}_3 : \text{mag\_}p^{\text{bf}}(Y) \leftarrow \text{mag\_}p^{\text{bf}}(X) \wedge e(X, Y)$$
$$\text{mar}_5 : \text{mag\_}p^{\text{bf}}(1).$$

5. Ước lượng bảng (Tabled Evaluation)

Ước lượng bảng là một phương pháp ước lượng các câu truy vấn trong cơ sở dữ liệu suy diễn, đã được nghiên cứu nhiều trong thời gian gần đây. (xem [2, 11]). Ước lượng bảng mở rộng khả năng của các ngôn ngữ lập trình logic vì nó có thể được dùng để ước lượng các câu truy vấn đệ quy như trong Prolog nhưng với các tính chất kết thúc tốt hơn nhiều. Phương pháp ước lượng bảng sẽ ngăn chặn các vòng lặp vô hạn (điều này thường xảy ra trong ước lượng SLD) và đảm bảo việc ước lượng sẽ kết thúc đối với chương trình Datalog.

Ý tưởng chính của phương pháp ước lượng bảng như sau: trong suốt quá trình ước lượng chương trình logic, các đích con và các câu trả lời được lưu giữ vào một bảng. Mỗi lời gọi đến đích con phải được kiểm tra xem đích con này (hoặc một biến thể của nó) được gọi trước đó hay không. Nếu không có thì đích con này được chèn vào bảng và các quy tắc được phân giải dựa vào đích con này y như trong phương pháp ước lượng SLD. Kết quả của việc ước lượng sẽ được đưa vào bảng. Nếu có một biến thể của đích con đã được gọi trước đó, đích con sẽ được phân giải dựa vào các câu trả lời đã có trong bảng. Các câu trả lời mới nhận được, đến lượt nó sẽ được thêm vào bảng và gắn liền với một đích con trong suốt quá trình ước lượng. Việc ước lượng sẽ kết thúc khi tất cả các quy tắc và các câu trả lời được phân giải nhờ vào việc áp dụng tất cả các đích con. Do các câu trả lời được tạo ra trong suốt tiến trình của cùng một ước lượng đã sử dụng chúng nên việc ước lượng

bảng có thể xem như một tính toán điểm bất động theo hướng bottom-up truyền thống. Việc ước lượng trên các đích con đạt tới một điểm cố định thì những đích con này gọi là được ước lượng đầy đủ.

Trong bài báo này, chúng tôi trình bày kỹ thuật ước lượng bảng bằng cách dùng các kí hiệu của phép phân giải SLG (Linear Resolution with Selection function for General logic programs) nhưng được xây dựng lại đơn giản hơn đối với các chương trình Datalog. Chi tiết về phép phân giải SLG đối với chương trình logic tổng quát có thể xem [2, 11].

5.1. Một số định nghĩa

Định nghĩa 5.1.1. Một bảng bao gồm một tập các đích con, mỗi đích con gắn liền với một tập các câu trả lời, hai đích con hoặc các câu trả lời được xem là đồng nhất với nhau trong bảng nếu chúng là các biến thể của nhau.

Định nghĩa 5.1.2. Một hệ thống SLG bao gồm một rừng cây SLG được xây dựng như sau:

- Một nút gốc của cây SLG có dạng: $q \leftarrow q$, trong đó q là một đích con. Nút gốc được gọi là *đầy đủ* khi cây tương ứng với nó là được ước lượng đầy đủ (xem Định nghĩa 5.1.5).
- Các nút không phải là nút gốc có một trong hai dạng:

  - Nút thất bại (Fail) được thêm vào như một nút lá của các nhánh thất bại, hoặc

  - Nút có dạng: Answer_Template ← Goal_List,

  trong đó Answer_Template là một nguyên tử được dùng để biểu diễn các trị buộc thay đổi của các đích con được xếp vào bảng trong suốt quá trình phân giải, Goal_List chứa danh sách các nguyên tử vẫn còn phải ước lượng. *Giả thiết quá trình tính toán trên các qui tắc được thực hiện từ trái sang phải, các nguyên tử được chọn của một nút là nguyên tử bên trái nhất trong Goal_List.*
- Một nút lá trong cây SLG được gọi là một nút trả lời nếu Goal_List của nó là rỗng.

Trong hệ thống SLG không có hai cây cùng nút gốc, nghĩa là các đích con tương ứng của chúng không thể là các biến thể đổi tên lẫn nhau.

Ước lượng câu truy vấn bằng cách dùng hệ thống SLG được gọi là một ước lượng SLG, nó được định nghĩa như sau.

Định nghĩa 5.1.3. Cho chương trình Datalog P, một ước lượng LSG **E** đối với một đích truy vấn *root* được xếp bảng là một dãy các rừng cây F_0, $F_1, \ldots, F_n$ sao cho:

- F_0 là rừng chứa một cây đơn *root* ← *root*.
- Với mỗi $i \geq 0$ hữu hạn, F_{i+1} nhận được từ F_i bằng cách áp dụng một trong các phép toán SLG (xem Định nghĩa 5.1.4). Nếu không có phép toán nào được áp dụng vào F_n thì F_n được gọi là hệ thống cuối cùng của ước lượng **E**.

Các cây mới sẽ được tạo bởi phép toán tạo *đích con mới* khi đích con được đưa bào bảng (là đích con mới cần tính) trở thành các literal được chọn của các nút. Gốc của các cây đôi khi còn được gọi là các nút tổ tiên. Phép toán *phân giải qui tắc* được dùng để sinh ra các nút con của các nút tổ tiên và các nút trong (nút trong là nút literal được chọn của nó là không được đưa vào bảng). Nếu literal được chọn của một nút là được đưa vào bảng thì các con của nó được sinh ra bởi phép toán *phân giải câu trả lời*. Các phép toán này được định nghĩa như sau.

Định nghĩa 5.1.4. (Các phép toán SLG)

(i) Đích con mới: Gọi N là nút không phải nút gốc: Answer_Template ← $A \wedge$ Goal_List trong đó đích con S được xếp vào bảng. Nếu S là đích con mới đối với việc ước lượng thì thêm một cây mới với gốc là qui tắc $S \leftarrow S$.

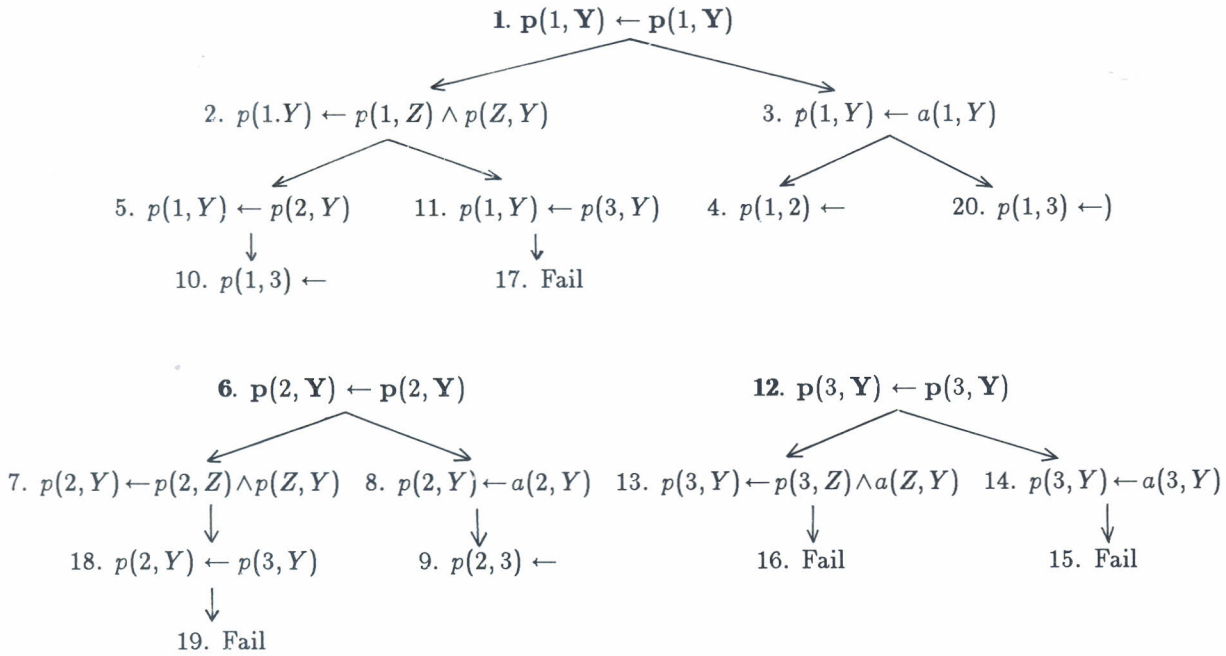
(ii) Phân giải qui tắc: Gọi N là một nút gốc $S \leftarrow S$ và R là qui tắc Head $\leftarrow$ Body, trong đó Head hợp nhất với đích con S với mgu là θ. Thêm nút $(S \leftarrow \text{Body})\theta$ như là một nút con của nút N nếu nó là nút mới.

Gọi N là nút không phải nút gốc: Answer_Template $\leftarrow S \wedge$ Goal_List, với S không xếp vào bảng. Gọi R là qui tắc Head $\leftarrow$ Body, trong đó đầu qui tắc Head hợp nhất với đích con S với mgu θ. Thêm nút (Answer_Template $\leftarrow$ Body $\wedge$ Goal_List)θ như là nút con của nút N.

(iii) Phân giải câu trả lời: Gọi N là nút không phải gốc mà literal S được chọn của nó là được xếp vào bảng và Ans là một nút trả lới. Gọi N' là hợp nhất của N và Ans trên S. Lúc đó nếu N' không phải là con của N thì thêm N' như là nút con của N.

(iv) Kết thúc: Cho trước một tập S các đích con được ước lượng đầy đủ (xem Định nghĩa 5.1.5), các nút gốc của các cây ứng với các đích con trong S được gọi là nút đầy đủ.

Ví dụ 5. Để minh họa các phép toán trong Định nghĩa 5.1.4, ta xét trở lại chương trình Datalog P trong Ví dụ 1, giả quan hệ A đối với vị từ e gồm các bộ $(1,2)$, $(1,3)$, $(2,3)$. Ước lượng SLG của chương trình P như sau:



Đích con	Trả lời
$p(1,Y)$	$p(1,2)$, $p(1,3)$
$p(2,Y)$	$p(2,3)$
$p(3,Y)$	

Chúng ta xem xét các phép toán SLG được áp dụng vào ước lượng SLG của ví dụ này. Trước tiên đích con $p(1,Y)$ được đưa vào bảng và hệ thống SLG khởi đầu từ nút 1. Các nút 6, 12 được tạo ra bằng cách dùng phép toán tạo đích con mới. Các nút 2, 3, 7, 8, 13 và 14 được tạo ra bằng phép toán phân giải qui tắc được áp dụng cho nút gốc. Các nút 4, 9, 20 được tạo ra bằng phép toán phân giải qui tắc từ các nút trong. Các nút 5, 11, 18 được tạo ra bằng phép toán phân giải câu trả lời. Trong hệ thống SLG này thì các nút 1, 6, 12 là các nút tổ tiên, các nút 3, 8, 14 là các nút trong, các nút trả lời là 4, 9, 10.

Định nghĩa 5.1.5. (Ước lượng đầy đủ)

Cho một hệ thống SLG và tập S các đích con được xếp vào bảng. S là được ước lượng đầy đủ nếu có ít nhất một trong các điều kiện sau đây được thỏa mãn đối với mỗi đích con $q \in S$:

1. q có một nút trả lời là một biến thể của q, hoặc
2. Đối với mỗi nút N trong cây có gốc là q:
   a) Literal được chọn SL của N là đầy đủ, hoặc
   b) $SL \in S$ và không còn áp dụng được các phép toán SLG đối với SL.

Tính đúng đắn của SLG đối với chương trình logic tổng quát được chỉ ra trong [11], từ đó dẫn theo tính đúng đắn trong trường hợp ta đang xét, là một thu hẹp của SLG đối với chương trình Datalog.

Gọi F là hệ thống LSG đối với ước lượng LSG của một chương trình P và câu truy vấn Q. Thể hiện bộ phận của F, $I(F)$, là một tập các nguyên tử cơ sở được xây dựng như sau: $A \in I(F)$ nếu A là một hiện hành cơ sở của một số câu trả lời trong F, $A \notin I(F)$ nếu A là hiện hành cơ sở của một vài nguyên tử A' và cây LSG đối với A' là đầy đủ trong F nhưng không chứa A như là một hiện hành của câu trả lời nào đó.

Ta có định lý sau đây:

Định lý. [11] *Gọi Q là câu truy vấn đối với chương trình Datalog P. Lúc đó một ước lượng SLG sẽ đạt đến một hệ thống cuối cùng F_n mà trong đó một nguyên tử cơ sở A là thuộc vào $I(F_n)$ nếu và chỉ nếu nó thuộc vào $M_{P/S}$, trong đó $M_{P/S}$ là mô hình cực tiểu của P được giới hạn đối với tập các đích con trong F.*

6. KẾT LUẬN

Việc phân tích tiến hành trong bài báo này bao gồm hai phương pháp chính nhằm ngăn chặn các vòng lặp vô hạn khi tìm kiếm các lời giải của câu truy vấn đối với chương trình Datalog. Cả hai phương pháp này thực chất thực hiện cùng một sự tính toán và đều là các thuật toán hướng đích. Sự khác nhau cơ bản trong việc thực hiện của cả hai cách tiếp cận này là về mặt cấu trúc dữ liệu. Phương pháp ước lượng bảng duy trì một cây stack của phép tính sao cho các câu trả lời được trả về trực tiếp đối với các phép toán. Với thuật toán ma tập, các câu trả lời được đưa ra bằng cách thực hiện phép toán nối. Mặc dù chiến lược ước lượng bảng được xem là thuộc cách tiếp cận top-down và phép biến đổi ma tập được xem là bottom-up, nhưng một điều đáng ghi nhận là ước lượng bảng giới thiệu một thành phần bottom-up trong khi phép biến đổi ma tập giới thiệu một thành phần top-down trong các chiến lược chung của chúng. Một điểm bất lợi của cả hai phương pháp là quá trình tìm câu trả lời truy vấn không tách rời được không gian tìm kiếm ra khỏi chiến lược tìm kiếm. Một số thuật toán chi tiết để ước lượng câu truy vấn đối với chương trình logic tổng quát có thể tìm thấy trong các tài liệu [2, 6, 11].

TÀI LIỆU THAM KHẢO

[1] Gallaire H., J. Minker, and J. M. Nicolas, Logic and Database: A Deductive Approach, Incomputing Survey, Vol. 16, 1984.

[2] J. Feire, T. Swift, D. S. Warren, Taking I/O seriously: resolution reconsidered for disk, *Proceeding of the International Conference on Logic Programming*, 1997.

[3] Krzysztof R. Apt, *Logic Programming*, Elsevier Science Publishers, 1990.

[4] Lê Mạnh Thạnh, An efficient Semi-Naive algorithm datalog, *Proceedings of the NCST of Vietnam* **11** (1999).

[5] Lloyd J. W., *Foundations of Logic Programming*, 2nd ed., Springer-Verlag, New York, 1987.

[6] R. Ramakrishnan, Magic templates: a spellbinding approach to logic programs, *Journal of Logic Programming* **11** (1991) 189–216.

[7] S. Ceri, G. Gottlob, L. Tanca, *Logic Programming and Databases*, Springer-Verlag, Berlin-Heidelberg, 1990.

[8] Serge Abiteboul, Richard Hull, Victor Vianu, *Foundation of Databases*, Addision Wesley Publishing Company, 1995.

[9] Subrata Kumar Das, *Deductive Databases and Logic Programming*, Addision Wesley Ed., 1992.

[10] Ullman J. D., *Principles of Database and Knowledge - Base Systems*, Computer Science Press, 1989.

[11] W. Chen and D. S. Warren, Tabled evaluation with delaying for general logic programs, *JACM* **43** (1) (1996) 20–74.

Nhận bài ngày 1 - 7 - 2001
Nhận lại sau khi sửa ngày 14 - 11 - 2001

Trường Đại học Khoa học, Đại học Huế.