MỘT PHƯƠNG PHÁP TÍNH MÔ HÌNH TRONG LẬP TRÌNH LUẬN LÝ

Trần Ngọc Thái Kha
Trường Đại học Bách Khoa – ĐHQG-HCM
(Bài nhận ngày 19 tháng 11 năm 2003)

TÓM TẮT: *Lập trình luận lý (Logic programmming) gồm có hai phần chính là ngữ nghĩa và giải thuật để tính mô hình từ chương trình cho trước. Hiện nay chưa có một giải thuật tổng thể để giải quyết bài toán này. Trong bài báo này, chúng tôi sẽ trình bày một số cải tiến trong việc tính mô hình đối với ngữ nghĩa tổng quát của lập trình luận lý. Giải thuật này phân chia chương trình thành các chương trình con, rút gọn và tính mô hình của các chương trình con và tổng hợp kết quả mô hình tổng thể dựa trên ngữ nghĩa Fitting, ngữ nghĩa Stable Model và giải thuật tìm "Thành phần liên kết chặt chẽ".*

Từ khóa: Lập trình luận lý, ngữ nghĩa, mô hình, chương trình.

I. Giới thiệu

Trong phần này chúng tôi giới thiệu các định nghĩa cơ bản của lập trình luận lý cũng như tóm tắt một số ngữ nghĩa và giải thuật phổ biến hiện nay.

I.1. Định nghĩa [1, 5]

- *Chương trình luận lý* gồm một tập hữu hạn các luật có dạng

$$A_0 \leftarrow A_1, ..., A_m, \neg A_{m+1}, ..., \neg A_n \quad (1)$$

với A_0 là nguyên tử đầu; A_1, ..., A_m là các nguyên tử thân dương và A_{m+1}, ..., A_n là các nguyên tử thân âm của luật.

Ví dụ:

$$giảm\_giá(A) \leftarrow học\_sinh(A), \neg\ kỷ\_luật(A)$$

Luật này có thể được đọc là: người A sẽ được giảm giá nếu người A là học sinh và không bị vi phạm kỷ luật trước đó.

Chương trình là đơn điệu nếu không có nguyên tử thân âm trong mọi luật. Ngược lại, chương trình là không đơn điệu.

- *Ngữ nghĩa* là các ý nghĩa và quy tắc của nguyên tử và luật được đưa ra để tính các mô hình của chương trình cho trước.

Ví dụ:

Nguyên tử có giá trị là Đúng, Sai hoặc KhôngBiết.

Giá trị của nguyên tử đầu sẽ là Đúng nếu tồn tại một luật mà các nguyên tử thân dương là Đúng và các nguyên sử thân âm là Sai.

Giá trị của nguyên tử đầu sẽ là Sai nếu với mọi luật mà các nguyên tử thân có giá trị Đúng hoặc Sai và tồn tại một nguyên tử thân dương là Sai hoặc một nguyên sử thân âm là Đúng.

- *Mô hình* là một tập các nguyên tử với một giá trị cụ thể nào đó đã được định nghĩa trong ngữ nghĩa và đảm bảo tất cả các luật của chương trình là đúng thực tế.

Ví dụ:

{học_sinh(an) = Đúng, kỷ_luật(an) = Sai, giảm_giá(an) = Đúng} là một mô hình nhưng {học_sinh(an) = Đúng, kỷ_luật(an) = Đúng, giảm_giá(an) = Đúng} không phải là mô hình.

- *Giải thuật* là phương pháp tính giá trị của các nguyên tử dựa trên ngữ nghĩa và chương trình cho trước. Giải thuật thường đi kèm với ngữ nghĩa cụ thể nào đó.

Ví dụ:

Với một tập các nguyên tử có giá trị cho trước, giá trị nguyên tử đầu sẽ được theo ngữ nghĩa trên. Quá trình này được lặp cho đến khi tập các nguyên tử có giá trị không đổi.

I.2. Ngữ nghĩa và giải thuật

Lập trình luận lý có quá trình phát triển rất lâu dài, đặc biệt trong những thập niên 80 và đầu 90, với nhiều ngữ nghĩa phổ biến như Fixpoint [5], Completeness [5], Fitting [2], Well-founded [3], Stable Model [4], ... Ngữ nghĩa đơn giản Fixpoint cho phép tính mô hình duy nhất của tất cả các chương trình đơn điệu. Đối với chương trình không đơn điệu, mỗi ngữ nghĩa chỉ giải quyết được một lớp các chương trình và thực tế việc xây dựng một ngữ nghĩa tổng quát là điều chưa thể. Chúng tôi giới thiệu hai ngữ nghĩa sẽ được đề cập trong giải thuật cải tiến sau này.

Cho chương trình P gồm tập các luật có dạng (1).

I.2.1. Ngữ nghĩa Fitting [2]

- *Ngữ nghĩa*

Giá trị của nguyên tử: Đúng, Sai, KhôngBiết.

$A_0 \in$ Đúng nếu $\exists\ A_0 \leftarrow A_1, ..., A_m, \neg A_{m+1}, ..., \neg A_n \in P$ mà

$\forall\ i \in [1, m]\ A_i =$ Đúng $\wedge\ \forall\ j \in [m+1, n]\ A_j =$ Sai

$A_0 \in$ Sai nếu $\forall\ A_0 \leftarrow A_1, ..., A_m, \neg A_{m+1}, ..., \neg A_n \in P$ mà

$(\forall\ i \in [1, n]\ A_i =$ Đúng $\vee\ A_i =$ Sai$) \wedge$

$(\exists\ j \in [1, m]\ A_j =$ Sai $\vee\ \exists\ k \in [m+1, n]\ A_k =$ Đúng$)$

- *Giải thuật*

B = {nguyên tử có giá trị Đúng hoặc Sai cho trước, nguyên tử không phải nguyên tử đầu có giá trị là Sai}

Repeat

A = B

Tính nguyên tử đầu theo ngữ nghĩa trên và tập giá trị B.

Until B = A

Ví dụ:

P_1 = {a ← ¬b. b ← ¬c.} có được một mô hình Fitting là {a = Sai, b = Đúng, c = Sai}

P_2 = {a ← ¬b. b ← ¬a.} không có mô hình Fitting. Thực tế, chương trình này có hai mô hình là {a = Đúng, b = Sai}, {a = Sai, b = Đúng}.

I.2.2. Ngữ nghĩa Stable Model [4]

- *Ngữ nghĩa*

Giá trị của nguyên tử: Đúng, Sai.

Cho một tập M các nguyên tử của chương trình P, gọi P_M là chương trình có được bằng cách xóa:

(i) Mỗi luật có nguyên tử thân âm ¬B với B ∈ M, và

(ii) Tất cả nguyên tử thân âm của các luật còn lại.

Rõ ràng P_M là chương trình đơn điệu và do đó nó có một mô hình duy nhất (Fixpoint). Nếu mô hình này trùng với M thì M là mô hình ổn định của P.

- Giải thuật

Đây là bài toán NP-complete nên nó không có lời giải cho bài toán tổng quát. Hiện nay, Simons đề nghị một giải thuật phức tạp tính mô hình ổn định dựa trên kinh nghiệm tỉa không gian tìm kiếm tập con M của tập tất cả các nguyên tử [7]. Chi tiết giải thuật nằm ngoài phạm vi bài báo này.

Ví dụ:

$P_2 = \{a \leftarrow \neg b.\ b \leftarrow \neg a.\}$ có hai mô hình Stable Model là {a = Đúng, b = Sai}, {a= Đúng, b = Đúng}.

$P_3 = \{a \leftarrow \neg b.\ b \leftarrow \neg a.\ c \leftarrow \neg c\}$ không có mô hình Stable Model. Tuy nhiên, trong thực tế, ta có hai mô hình {a = Đúng, b = Sai}, {a = Sai, b = Đúng}.

II. Giải thuật cải tiến

Giải thuật cải tiến dựa chủ yếu trên tư tưởng của hai ngữ nghĩa trên. Ngữ nghĩa Fitting được sử dụng để rút gọn và tính giá trị các nguyên tử (nếu được). Chúng tôi cũng biểu diễn chương trình thành đồ thị có hướng với nút là nguyên tử và cạnh có hướng từ nguyên tử thân (âm hoặc dương) đến nguyên tử đầu. Các đồ thị con được tách rời và sắp xếp lại thứ tự theo tính chất của "Thành phần liên kết chặt chẽ". Các chương trình con được tách ra tương ứng với các đồ thị con trên. Với từng chương trình con, chúng tôi sẽ tính mô hình ổn định, thêm và cập nhật vào mô hình tổng thể.

Giải thuật:

```
M[1] = khởi_tạo_mô_hình(∅)
P' = rút_gọn_tính_Fitting(P, M[1])
G = biểu_diễn_đồ_thị_có_hướng(P')
SG[] = tách_sắp_xếp_thành_phần_liên_kết_chặt_chẽ(G)
For mỗi đồ thị con SG[i] Do
  SP[i] = tách_chương_trình_con(SG[i]);
  For mỗi mô hình M[j] Do
      SP'[i] = rút_gọn_tính_Fitting(SP[i], M[j])
      SM[] = tính_mô_hình_ổn_định(SP'[i])
      For mỗi mô hình ổn định SM[k] Do
          thêm_cập_nhật_mô_hình(SM[k], M[j])
      End For
  End For
End For
```

II.1. Thành phần liên kết chặt chẽ [6]

Khuyết điểm của ngữ nghĩa Stable Model là nếu một phần chương trình không có mô hình thì toàn bộ chương trình sẽ không có mô hình. Đều này hoàn toàn trái ngược với thực tế vì chúng không liên quan với nhau. Ý tưởng ở đây là chúng tôi sẽ tách chương trình thành những chương trình con độc lập hoặc chỉ liên kết theo một hướng có thứ tự xác định. Rõ ràng, nếu điều này có thể thực hiện được thì khuyết điểm của ngữ nghĩa Stable Model sẽ được khắc phục. Hơn nữa, việc tính toán mô hình ổn định sẽ đơn giản hơn rất nhiều.

"Thành phần liên kết chặt chẽ" trong lý thuyết đồ thị có hướng cho phép chúng ta tách đồ thị thành từng phần độc lập hoặc có hướng xác định. May mắn là việc biểu diễn chương trình dưới dạng đồ thị có hướng có thể được thực hiện với nút là các nguyên tử và cạnh có hướng từ nguyên tử thân (âm hoặc dương) đến nguyên tử đầu của tất cả các luật trong chương trình. Hiện nay có rất nhiều giải thuật tính đồ thị con "Thành phần liên kết chặt chẽ", tuy nhiên chúng tôi sử dụng giải thuật đơn giản và kinh điển của Tarjan.

Việc thao tác trên đồ thị con (chứa nút gốc) sẽ ảnh hưởng đến đồ thị con (chứa nút đỉnh) có cạnh nối giữa hai đồ thị con này và hướng từ nút gốc đến nút đỉnh. Dựa trên tính chất có hướng xác định, các đồ thị con được sắp xếp theo thứ tự sau: Với hai đồ thị con có liên kết với nhau thì đồ thị con chứa nút gốc sẽ có thứ tự trước đồ thị con chứa nút đỉnh. Điều này đảm bảo các mô hình của các chương trình con tương ứng với các đồ thị con sẽ chỉ tính một lần duy nhất.

Chương trình con được tách dựa vào đồ thị con với quy tắc sau: Nếu các nguyên tử đầu và các nguyên tử thân (âm và dương) thuộc đồ thị con thì luật này sẽ thuộc chương trình con.

II.2. Fitting mở rộng

Để mô tả ý nghĩa của các chương trình con không có mô hình, chúng tôi đề nghị tập giá trị của nguyên tử gồm {Đúng, Sai, KhôngBiết, KhôngXácĐịnh}. Giá trị KhôngXácĐịnh sẽ biểu diễn cho các nguyên tử trong chương trình con không có mô hình.

Ngữ nghĩa Fitting cũng sẽ được mở rộng như sau:

$A_0 \in$ Đúng nếu $\exists\ A_0 \leftarrow A_1, ..., A_m, \neg A_{m+1}, ..., \neg A_n \in P$ mà

$\forall\ i \in [1, m]\ A_i =$ Đúng $\wedge\ \forall\ j \in [m+1, n]\ A_j =$ Sai

$A_0 \in$ Sai nếu $\forall\ A_0 \leftarrow A_1, ..., A_m, \neg A_{m+1}, ..., \neg A_n \in P$ mà

$(\forall\ i \in [1, n]\ A_i =$ Đúng $\vee\ A_i =$ Sai $\vee\ A_i =$ KhôngXácĐịnh$) \wedge$

$(\exists\ j \in [1, m]\ A_j =$ Sai $\vee\ A_j =$ KhôngXácĐịnh$) \vee$

$(\exists\ k \in [m+1, n]\ A_k =$ Đúng $\vee\ A_k =$ KhôngXácĐịnh$)$)

Ở đây, chương trình rút gọn có được bằng cách xóa:

(i) Luật trong P mà nguyên tử đầu có giá trị Đúng, Sai hoặc KhôngXácĐịnh

(ii) Luật trong P mà nguyên tử thân dương có giá trị Sai hoặc KhôngXácĐịnh

(iii) Luật trong P mà nguyên tử thân âm có giá trị Đúng hoặc KhôngXácĐịnh

(iv) Nguyên tử thân dương có giá trị Đúng

(v) Nguyên tử thân âm có giá trị Sai

II.3. Stable Model từng phần

Chương trình đã rút gọn chỉ còn tập các luật với các nguyên tử chưa có giá trị thuộc Đúng, Sai hoặc KhôngXácĐịnh (nghĩa là mang giá trị khởi tạo KhôngBiết). Phần chương trình còn lại chính là khuyết điểm đã được đề cập của ngữ nghĩa Fitting. Chúng ta sẽ sử dụng ngữ nghĩa Stable Model để giải quyết phần bù này. Một điểm khá thú vị của giải thuật này chính là tính phối hợp và ý nghĩa dùng lại các giải thuật phổ biến trong việc giải quyết các khuyết điểm của chính các giải thuật này.

Việc sử dụng "Thành phần liên kết chặt chẽ" giúp chúng ta khắc phục được khuyết điểm của ngữ nghĩa Stable Model khi tính các phần chương trình độc lập. Đối với các phần chương trình liên quan, nếu phần chương trình thứ tự trước có mô hình ổn định thì phần sau được tính bình thường dựa trên kết quả trước. Ngược lại, nếu phần chương trình trước không có mô hình, Fitting mở rộng (thêm một giá trị KhôngXácĐịnh) sẽ giúp xóa các liên kết giữa các

phần chương trình này. Do đó phần chương trình thứ tự sau vẫn sẽ tính được mô hình của mình.

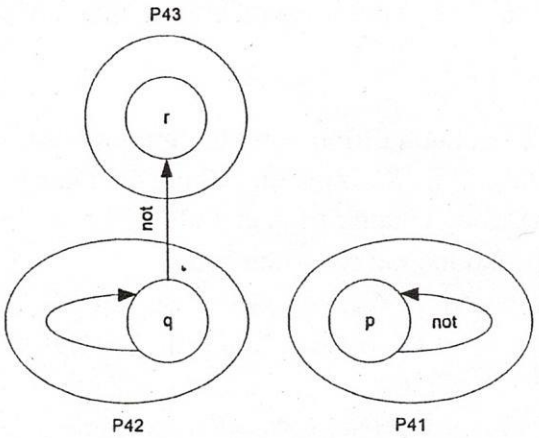
II.4. Nhận xét

Với những cải tiến trên, giải thuật cho phép tính mô hình rất tốt. Rõ ràng, nó khắc phục được các khuyết điểm trong ngữ nghĩa Fitting, Stable Model.

II.5. Ví dụ

$P_4 = \{p \leftarrow \neg p.\ q \leftarrow q.\ r \leftarrow \neg q.\}$

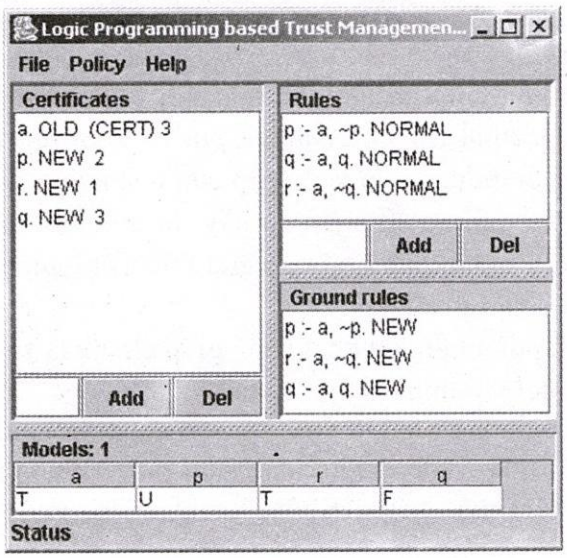
- *Fitting*: không mô hình.
- *Stable Model*: không mô hình vì kết quả tính mô hình của luật $p \leftarrow \neg p.$ làm ảnh hưởng mô hình tổng thể.
- *This algorithm*:



Chương trình sẽ được tách thành ba chương trình con theo thứ tự: $P_{41} = \{p \leftarrow \neg p.\}$, $P_{42} = \{q \leftarrow q.\}$ và $P_{43} = \{ r \leftarrow \neg q.\}$. P_{41} không có mô hình, P_{42} có một mô hình {q = Sai} và P_{43} có một mô hình {r = Đúng}. Mô hình tổng thể là {p = KhôngXácĐịnh, q = Sai, r = Đúng}.

II.6. Chương trình

Hiện tại chúng tôi đang cũng đang phát triển ứng dụng Logic Programming based Trust Management 1.1.3 dựa trên giải thuật cải tiến trên. Kết quả của giải thuật là rất khả quan.



III. Kết luận và đề nghị

Giải thuật này chính là sự phối hợp hai ngữ nghĩa phổ biến Fitting và Stable Model với sự hỗ trợ của tính chất "Thành phần liên kết chặt chẽ" trong lý thuyết đồ thị. Nó phát huy thế mạnh của từng ngữ nghĩa cũng như khắc phục khuyết điểm lẫn nhau. Tuy nhiên, giải thuật chỉ dừng lại ở mức thực nghiệm. Trong thời gian tới, chúng tôi sẽ khảo sát tính cần thiết và đầy đủ một cách tổng quát dựa trên mô hình toán học giải thuật này.

AN ALGORITHM FOR CALCULATING MODELS IN LOGIC PROGRAMMING

Tran Ngoc Thai Kha

ABSTRACT: *Logic programming consists of two main components (semantic and algorithm) in order to calculate the models from particular logic programs. Recently, there is not an algorithm for solving this problem completely. In this paper, we will present an extended algorithm for calculating models of general semantic in logic programming. The extended algorithm divides the originally program into subprograms, reduces, calculates subprogram models and then synthetize to the whole program models. It is based on fitting semantic, stable model semantic and "strongly connected components".*

TÀI LIỆU THAM KHẢO

[1] Baral, C., *Knowledge representation, reasoning and declarative problem solving with Answer Set*, Cambridge University Press, 2003.

[2] Fitting, M., A Kripke-Kleene semantics for Logic Programs, *Journal of Logic Programming*, Volumn 2, Number 4, 295-312, 1985.

[3] Gelder, A. V., K. A. Ross và J. S. Schlipf, The Well-Founded Semantics for General Logic Programs, *Journal of the ACM*, Volumn 38, Number 3, 620-650, 1991.

[4] Gelfond, M. và V. Lifschitz, The Stable Model Semantics for Logic Programming, *Proceedings of the 5th International Conference Symposium on Logic Programming*, 1070-1080, 1988.

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

[6] Sedgewick, R., *Algorithms*, Addison-Wesley, 1998.

[7] Simons, P., Extending and Implementing the Stable Model Semantics, *Research report* 58, Helsinki University of Technology, 2000.