TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN TẬP V (1989), Số 3 (24–28)

PHÉP CHUYỂN TỪ ĐẶC TẢ ẨN TÀNG THÀNH CHƯƠNG TRÌNH LOGIC

DƯƠNG TUẤN ANH

Một đặc tả ban đầu thường ẩn tàng và không khả thi, do đó cần thiết phải biến đổi nó về dạng khả thi. Bài báo này trình bày cách chuyển một đặc tả ban đầu thành một chương trình PROLOG bằng cách đưa vào đệ quy và thông qua một bước tinh chế dữ liệu. Sự biến thể này đóng góp vào một phương pháp luận lập trình nhằm dẫn ra chương trình từ đặc tả hình thức.

Trong phần I sơ lược về phương pháp đặc tả theo tiền/hậu điều kiện của C. B. Jones và trong phần II trình bày những nguyên tắc chuyển một đặc tả ẩn tàng thành những đặc tả tường minh bằng cách tổng hợp các hàm đệ quy và cách thức diễn tả các kiểu trừu tượng bằng kiểu ngôn ngữ PROLOG.

I – ĐẶC TẢ HÌNH THỨC THEO TIỀN/HẬU ĐIỀU KIỆN

C. B. Jones [2] đã đưa ra một phương pháp đặc tả theo tiền/hậu điều kiện (T/H ĐK) khá hoàn chỉnh để đặc tả bài toán ở giai đoạn đầu của công tác lập trình. Trong số những phương pháp đặc tả đã phát triển bởi các phương pháp lập trình, đây là phương pháp trực tiếp và trực giác nhất [3].

Đặc trưng chung của một đặc tả hình thức là:

– Nó độc lập đối với ý tưởng thực thi.

– Nó trình bày chính xác những tính chất của kết quả phải tìm mà không hề đưa ra cách thức để đạt đến, người ta bảo nó là *ẩn tàng*.

– Nó dùng một ngôn ngữ giả toán học, gần với ngôn ngữ kiểu tập hợp và phép toán tân từ.

Một đặc tả T/H ĐK cho một kiểu dữ liệu trừu tượng bao gồm:

1) Sự mô tả các đối tượng trừu tượng theo các đối tượng cơ sở (tập hợp, ánh xạ, danh sách, bộ đa thành phần), các kiểu tiền định (nguyên, lôgic) và các kiểu tham số.

2) Sự mô tả các thao tác làm việc trên kiểu trừu tượng sẽ gồm việc nêu lên:

– Các miền xác định và các miền giá trị.

– Những hạn chế về miền xác định của chúng, đặc trưng bằng các tiền điều kiện.

– Sự đặc trưng hóa kết quả bằng hậu điều kiện. C. B. Jones [2] đưa ra một hệ ký hiệu đầy đủ gồm bốn lối ký hiệu: tập hợp, ánh xạ, danh sách và cú pháp trừu tượng, làm thành một ngôn ngữ giả toán học thuận lợi trong việc đặc tả. Ngôn ngữ đặc tả ấy được thể hiện trong thí dụ sau đây mà được dùng trong suốt bài này.

Thí dụ: Đặc tả bài toán ghi nhận được quan hệ tương đương. Một quan hệ tương đương định nghĩa trên một tập phần tử (tên Element) sẽ hình thành một phân hoạch trên tập này mà mỗi khối là một lớp tương đương. Kiểu trừu tượng ở đây là phân hoạch. Ý tưởng cơ bản là diễn tả phân hoạch bằng một ánh xạ tương ứng với một phần tử với lớp tương đương mà nó thuộc về. Như vậy, ta khai báo kiểu trừu tượng

$$Qm = Element \rightarrow Element\text{-}set$$

Hai thao tác làm việc trên kiểu trừu tượng Qm là INIT khởi tạo một phân hoạch ban đầu và EQUATE biến đổi phân hoạch ban đầu thành phân hoạch kết quả (tập hợp các lớp tương

đương muốn tìm) căn cứ vào các cặp phần tử tương đương. Đặc tả hình thức của bài toán như sau:

Pair :: X : Element Y : Element
Block = Element − set
Qm = Element → Block
INIT
states : Qm
post − INIT (qm, qm') $\triangleq$ qm' = [e → {e} | e∈ Element]
EQUATE
states: Qm
type: Pair − Set →
post − EQUATE (qm, ps, qm') $\triangleq$ qm' = equate (qm, ps)
equate: Qm Pair − Set → Qm
equate (qm, ps) $\triangleq$ if ps = { } then qm
else let e∈ps equate (equ(qm, e), ps − {e})
equ : Qm Pair → Qm
equ (qm, e) $\triangleq$ if qm (X(e)) = qm(Y(e)) then qm
else let b = qm(X(e)) ∪ qm(Y(e))
qm + [X(e) → b, Y(e) → b]

Trong đặc tả trên, ta đã đưa vào hai hàm phụ equate và equ để hỗ trợ cho việc mô tả thao tác EQUATE.

II − PHÉP CHUYỂN ĐẶC TẢ ẨN TÀNG THÀNH CHƯƠNG TRÌNH PROLOG

Một đặc tả ẩn tàng chỉ cung cấp một cái nhìn ngoại tại về hệ thống, nói lên nó làm cái gì chứ không nói nó làm như thế nào. Từ đó, phải thực hiện nhiều bước biến thể được gọi là tinh chế dữ liệu để đi đến chương trình cuối cùng. Ở mỗi bước tinh chế dữ liệu, kiểu dữ liệu trừu tượng được thi công bằng những kiểu dữ liệu cụ thể hơn. Điều này được tiếp diễn cho đến khi kiểu dữ liệu trong đặc tả đồng nhất với các kiểu có sẵn ở ngôn ngữ lập trình và khi đó giải pháp của bài toán hoàn toàn xuất hiện. lợi dụng tính chất chương trình PROLOG là một đặc tả khả thi, ta có thể dùng nó như là một chặng trung gian trong quá trình biến đổi đặc tả ban đầu thành chương trình.

Khác với đặc tả theo T/H ĐK, chương trình PROLOG là một đặc tả hình thức có hàm chứa giải pháp của bài toán dưới dạng đệ quy và PROLOG chỉ cho phép diễn đạt các đối tượng trừu tượng theo 2 loại đối tượng cơ sở là danh sách và bộ đa thành phần. Vì vậy, sự biến thể bao gồm hai bước chuyển:

1. Phép chuyển ẩn tàng thành tường minh

Trong đặc tả T/H ĐK, ta thường dùng các tân từ của logic tân từ bậc nhất để diễn tả mối liên hệ giữa các dữ liệu vào và kết quả cũng như ý nghĩa của các hàm phụ. Chẳng hạn, để xét một danh sách l có thứ tự không, ta dùng hàm phụ is − ordered được mô tả như sau:

is − ordered (l) $\triangleq$ ($\forall$ i, j∈ [1 : len l]) (i<j ⇒ l(i) ≺ l(j))

Sự mô tả này là ẩn tàng. Bằng cách đưa vào đệ quy trên cấu trúc danh sách ta định nghĩa lại is − ordered như sau:

is − ordered (l) $\triangleq$ if : l = ⟨ ⟩ then true
else hd l ≼ hd td tl l ∧ is − ordered (tl l)

Đấy là một đặc tả tường minh, có hàm ý rằng để xét một danh sách có thứ tự ta lần lượt so sánh hai phần tử kế cận nhau.

Bởi đệ quy là một công cụ khái niệm quan trọng để thiết kế thuật toán, từ đặc tả ẩn tàng, ta phải tổng hợp thành những định nghĩa đệ quy chính thống. Đáng quan tâm nhất là sự tổng hợp các định nghĩa đệ quy trên tập hợp và danh sách.

Dạng thức cơ bản của một hàm đệ quy:

Dạng 1: (định nghĩa trên tập hợp):

f(S,x_1, x_2,...,x_n) $\triangleq$ if S = { } then g(x_1, x_2, ..., x_n) else
let e ∈ S
h(x_1,...,x_n, e, S − {e}, f(e,x_1,...,x_n), f(S − {e}, x_1,...,x_n)

Dạng 2: (định nghĩa trên danh sách):

$f(l, x_1, x_2,...,x_n) \triangleq$ if $l = \langle\ \rangle$ then $g(x_1, x_2,...,x_n)$ else

$h(x_1, ...x_n, hd\ l, tl\ l, f(hd l, x_1,...,x_n), f(tl l, x_1,...,x_n)$

Phương pháp đặc tả T/H ĐK thường cho phép dùng các biểu thức tập hợp trong các định nghĩa. Bằng cách đưa đệ quy vào, ta có thể loại trừ các biểu thức tập hợp không khả thi ra khỏi đặc tả.

J. Darlington [4] đã đưa ra một số qui tắc ước lược và qui tắc suy diễn tiện dụng cho việc tổng hợp đệ quy trên tập hợp.

a) *Các qui tắc ước lược cơ bản*

(i) Dạng 1: $\{f(x) \mid P(x) \vee Q(x)\}$

R1: $\{f(x) \mid P(x) \vee Q(x)\} \Leftarrow \{f(x) \mid P(x)\} \cup \{f(x) \mid Q(x)\}$

Dấu $\Leftarrow$ trong s1 $\Leftarrow$ s2 diễn tả biểu thức tập hợp s2 có thể giản lược về s1.

(ii) Dạng 2: $\{f(x) \mid x \in S \wedge P(x)\}$

R2a: $\{f(x) \mid x \in \{\} \wedge P(x)\} \Leftarrow \{\}$

R2b: $\{f(x) \mid x \in \{s\} \cup S \cap P(x)\}$

$\Leftarrow$ if P(s) then $\{f(s)\} \cup \{f(x) \mid x \in S \wedge P(x)\}$

eles $\{f(x) \mid x \in S \wedge P(x)\}$

R2c: $\{f(x) \mid x \in S_1 \cup S_2 \wedge P(x)\} \Leftarrow \{f(x) \mid x \in S_1 \wedge P(x)\} \cup \{f(x) \mid x \in S_2 \wedge P(x)$

(iii) Dạng 3: $\{f(X) \mid X \subseteq S \cap P(X)\}$

R3a: $\{f(X) \mid X \subseteq \{\} \wedge P(X)\} \Leftarrow$ if $P(\{\})$ then $\{\{\}\}$ else $\{\}$

R3b: $\{f(X) \mid X \subseteq \{s\} \cup S \wedge P(X) \Leftarrow \{f(X) \mid X \subseteq S \wedge P(X)\} \cup \{f(\{s\} \cup X) \mid X \subseteq S \wedge P(\{s\} \cup X)\}$

R3c: $\{f(X) \mid X \subseteq S_1 \cup S_2 \wedge P(X)\} \Leftarrow \{f(X_1 \cup X_2) \mid X_1 \subseteq S_1, X_2 \subseteq S_2 \wedge P(X_1 \cup X_2)\}$

b) *Các qui tắc suy diễn*

Cho trước một định nghĩa có chứa biểu thức tập hợp, để phát sinh ra những định nghĩa mới có khả năng dẫn đến hàm đệ quy, ta dùng một số qui tắc suy diễn, trong đó quan trọng nhất là qui tắc khai triển và qui tắc xếp gọn.

Định nghĩa 1 (Qui tắc khai triển).

Nếu $E \Leftarrow E'$ và $F \Leftarrow F'$ là các phương trình và có xuất hiện ở F' một thể hiện của E, hãy thay nó bằng thể hiện tương ứng của E' và do đó đạt được phương trình mới $F \Leftarrow F''$.

Định nghĩa 2 (Qui tắc xếp gọn).

Nếu $E \Leftarrow E'$ và $F \Leftarrow F'$ là các phương trình và có xuất hiện trong F' một thể hiện của E', hãy thay nó bằng thể hiện của E, ta được một phương trình mới $F \Leftarrow F''$.

Thí dụ:

Chuyển predset $(S,p) \triangleq \{x \in S \mid p(x)\}$ về dạng đệ quy.

1. predset $(S, p) \triangleq \{x \in S \mid p(x)\}$
2. predset $(\{\}, p) \Leftarrow \{x \in \{\} \mid p(x)\}$ (do quy tắc khai triển)
   $\Leftarrow \{\}$ (do R2a)
3. predset $(\{e\} \cup S, p) \Leftarrow \{x \in \{e\} \cup S \mid p(x)\}$ (do qui tắc khai triển)
4. $\Leftarrow$ if p(e) then $\{e\} \cup \{x \in S \mid p(x)\}$
   else $\{x \in S \mid p(x)\}$ (do R2b)
5. $\Leftarrow$ if p(e) then $\{e\} \cup$ predset (S,p)
   else predset (s, p) (do qui tắc xếp gọn)

Phối hợp 2 và 5 ta được hàm đệ quy:

predset $(S, p) \triangleq$ if $S = \{\}$ then $\{\}$

else let $e \in S$ if p(e) then $\{e\} \cup$ predset $(S - \{e\}, p)$

else predset $(S - \{e\}, p)$

2. Bước tính chế dự liệu

Sự mô tả các đối tượng dữ liệu ở PROLOG chỉ có thể thông qua danh sách và bộ đa thành phần. Để hướng tới PROLOG, ta phải thực hiện một số bước tinh chế dữ liệu đầu tiên mà trong đó những kiểu trừu tượng nào trong đặc tả ban đầu được diễn tả trừu tượng theo tập hợp và ánh xạ thì diễn tả lại theo danh sách.

i) Khi dùng danh sách diễn tả tập hợp, toán tử $\cup$ được chuyển thành || hay unionl, $\cap$ thành intersectl, $\setminus$ thành diffl và hàm phụ predset được chuyển thành hàm phụ predlist: unionl $(l_1, l_2) \triangleq$ if $l_1 = \langle\ \rangle$ then l_2

else if hd l_1 ∈ elems l_2 then unionl (tl l_1, l_2)

else < hd l_1 > || unionl (tl l_1, l_2)

diffl (l_1, l_2) Δ if l_1 = < > then < >

else if hd l_1 ∈ elems l_2 then diffl (tl l_1, l_2)

else < hd l_1 > || diffl (tl l_1, l_2)

intersectl (l_1, l_2) Δ if l_1 = < > then < >

else if hd l_1 ∈ elems l_2 then < hd l_1 > ||

intersectl (tl l_1, l_2) else intersectl (tl l_1, l_2)

predlist (l_1, p) Δ if l_1 = < > then < >

else if p (hd l_1) then < hd l_1 > || predlist (tl l, p)

else predlist (tl l, p)

ii) Đối với những kiểu trừu tượng được diễn tả bằng ánh xạ, ta có thể dùng danh sách cặp – phần – tử để diễn tả.

Bước tinh chế dữ liệu theo những nguyên tắc nói trên thường làm phát sinh ra nhiều hàm phụ trợ mà cũng là hàm đệ quy, để thực hiện một thao tác trong đặc tả ban đầu. Các hàm phụ trợ này chi tiết hóa dần dần giải pháp của bài toán. Điều đó phù hợp với triết lý phân rã một thao tác thành nhiều thao tác nhỏ đơn giản hơn.

Sau bước phát triển thứ nhất, đặc tả ở hình 1 trở thành đặc tả ở hình 2.

Đến đây, ta đạt được một đặc tả tường minh, dùng nhiều định nghĩa đệ quy và có bao hàm một giải pháp cho bài toán. Từ đó việc chuyển thành một chương trình logic là một bước trực tiếp, miễn là ta chú ý:

– Mỗi định nghĩa đệ quy dùng if then else được tách thành nhiều mệnh đề định nghĩa đệ quy theo trường hợp.

– Phép hợp hàm ở đặc tả T/H–ĐK được chuyển thành một dãy tân từ nối nhau bằng toán tử and.

Đặc tả bài toán qua hệ tương đương sau bước phát triển thứ nhất:

Pair: X: Element Y: Element

Block 1 = Element – list

Partiton = Rlock 1 – list

INIT 1

states: Partion

post–INIT 1 (pn, $p^{n'}$) Δ $p^{n'}$ = init (Element)

EQUATE 1

states: Partition

type: Pair – list →

post – EQUATE 1 (pn, pl, p'_n) Δ p'_n = equate (pn, pl)

equate: Partition Pair–list → Partition

equate (pn, pl) Δ if pl = < > then ph

else equate (equ (pn, hd pl), tl pl)

equ: Partition Pair → Partiiion

equ (pn, e) Δ let a = class (pn, X(e)), let b = class (pn, Y(e)).

if a = b then pn else let c = a || b

delete (delete (pn, a), b) || c

init: Element–list → Partition

init (*l*) $\triangleq$ if *l* = <> then <> else ≪hdl≫ || init (tle)

class: Partition Element → Block 1

class (pn, e) $\triangleq$ if e ∈ elems (hd pn) then hd pn
else class (tl pn, e)

delete: Partition Block 1 → Partition

delete (pn, b) $\triangleq$ if b = hd pn then tl pn else
hd pn || delete (tl pn, b)

Từ đặc tả trên, ta đi đến chương trình PROLOG dưới đây trong đó có các tân từ member, append, delete có thể coi như các hàm thư viện có sẵn, không cần định nghĩa thêm.

domains

list = symbol*
parti = list*
pair = p (symbol, symbol)
pairlist = pair*

predicates

run (list, pairlist, parti)
init (list, parti)
equate (parti, pairlist, parti)
equ (parti, pair, parti)
class (symbol, parti, list)

clauses

run (EL, PL, Res) ← init (L, IP), equate (IP, PL, Res)
equate (PL, [ ], PL).
equate (PL, [p (X, Y)] T, Res) ← equ (PL, p(X,Y), PN1), equate (PN1, T, Res).
equ (PL, p (X, Y), PL) ← class (X, PL, A), class (Y, PL, A).
equ (PL, p (X, Y), [C | PL2] ← class (X, PL, A), class (Y, PL, B),
delete (PL, A, PL1), delete (PL1, B, PL2), append (A, B, C).
class (X, [H | −], H) ← member (X, H).
class (X, [− | T], Z) ← class (X, T, Z).
init ([X], [[X]]).
init ([X | Y], [X] | [Y]) ← init (Y, LY).

KẾT LUẬN

PROLOG không chỉ là một công cụ ngôn ngữ rất quan trọng trong các áp dụng trí tuệ nhân tạo ([1], [5]) mà còn có thể dùng làm một mức đặc tả trung gian trong quá trình phát triển chương trình từ đặc tả ban đầu. Việc có thể chạy thử trên máy tính (có sẵn hệ lập trình PROLOG) mức đặc tả này làm cho lập trình viên sớm có thể kiểm tra tính đúng đắn của những ý tưởng thiết kế ban đầu. Điều đó rất có ý nghĩa trong việc tạo ra những phần mềm tin cậy.

TÀI LIỆU THAM KHẢO

1. **Kowalki R.**, Algorithm = Logic + Control, Comm. ACM, Vol. 22, No 7, June 1979, pp 424 − 435.

2. **Jones, C.B.**, Software development: A Rigorous approach, Prentia Hall International, INC, London, 1980.

3. **Derniame. J.C.**, Finance. J. P., Types Abstrait de données: Specification, Utilissation et realization, Ecole d'éte de l' AFCET, 1979.

4. **Darlington. J.**, Application of Program Transformation to program Synthesis in Proving and Improving program, Collogues IRIA, Arc et Senancs, July 1975, pp. 133 − 144.

5. **Borland.**, JMC, IBM, Turbo Prolog: The Natural language of Artificial Intelligence, 1986.

A Way to transform implicit specifications in to logic programs

This paper presents a way to transform an initial specification of a problem into a PROLOG program by introducing recursion and through a data retinement step. This transformation contributes to a programming methodology which intends to derive programs from formal specifications.