TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN, Tập III (1987), Số 3 (9–15)

CÀI ĐẶT NGÔN NGỮ PADRE BẰNG KỸ THUẬT TIỀN XỬ LÍ

NGUYỄN VĂN LỰ, *ĐHTH Pie Quyri và Mari Quyri.*
HỒ THUẦN, *Viện Khoa học tính toán và điều khiển.*

TÓM TẮT

Trong bài báo này những bước khác nhau và những ưu điểm của kỹ thuật tiền xử lí đã được chỉ ra thông qua cài đặt cụ thể ngôn ngữ PADRE, được thiết kế nhằm mô tả, biến đổi và thể hiện những lớp mạng Petri khác nhau. Ở đây chúng tôi cũng chứng tỏ rằng kỹ thuật tiền xử lí này hoàn toàn có thể áp dụng được trên máy vi tính.

1. MỞ ĐẦU

Trong [1] đã chứng tỏ rằng PADRE là ngôn ngữ lập trình thử nghiệm có thể dùng để mô tả, biến đổi và thể hiện những lớp mạng Petri khác nhau, chẳng hạn mạng Petri thông thường, mạng Petri có trọng số ([2]), mạng Petri màu ([3]), mạng Petri tân từ/chuyền ([1]),... Trong bài báo này sẽ trình bày một kỹ thuật cài đặt cụ thể ngôn ngữ Padre.

Với cách cài đặt này chúng tôi nhằm hai mục đích chính là tốc độ thực hiện và khả năng trao chuyền. Mục đích thứ nhất có thể đạt được nhờ kỹ thuật tiền xử lí nhằm dịch từ ngôn ngữ nguồn cấp cao sang ngôn ngữ đích cấp cao khác. Vì ngữ nghĩa của các ngôn ngữ nguồn và đích thường dễ hiểu, nên quá trình dịch không có nhiều khó khăn lắm do có sự tương đương trực tiếp về ngữ nghĩa giữa chúng với nhau. Nhằm nâng cao khả năng trao chuyền, chúng tôi đã chọn một ngôn ngữ đã được cài đặt trên phần lớn tất cả các họ máy tính, đó là ngôn ngữ Pascal. Luôn hướng tới hai mục đích này, kỹ thuật tiền xử lí được đưa ra bao gồm 4 bước: (i) Tạo sinh các biểu diễn trong có các kiểu khác nhau cùng với những thông tin thích hợp để dùng về sau, (ii) dịch toàn bộ chương trình viết bằng ngôn ngữ PADRE sang chương trình viết bằng ngôn ngữ PASCAL tương đương về ngữ nghĩa, (iii) tạo sinh các thủ tục tiện ích đáp ứng nhu cầu của người sử dụng (iv) chuyển những thông tin ra ở các bước trước cho chương trình dịch PASCAL để hoàn thành quá trình dịch. Tất nhiên, thiết kế bộ tiền xử lý phải bao gồm tất cả những thuật toán chính quen biết trong quá trình dịch ([5]) như phân tích từ vựng và cú pháp, phân tích ngữ nghĩa, phát hiện lỗi,... chỉ có điều là mã đối tượng được tạo sinh dưới dạng ngôn ngữ cấp cao

Bài báo chia làm 3 phần. Trong phần 1 sẽ nhắc lại một vài cấu trúc cơ bản của ngôn ngữ PADRE, phần 2 mô tả kỹ thuật cài đặt, còn trong phần 3 chúng tôi sẽ chứng tỏ rằng kỹ thuật này hoàn toàn có thể dùng cho máy vi tính.

2. SƠ LƯỢC ĐÔI NÉT VỀ NGÔN NGỮ PADRE

Trong phần tóm tắt về ngôn ngữ PADRE được đưa ra sau đây chúng tôi chỉ nhắc lại một vài cấu trúc điển hình của ngôn ngữ này. Ngoài ra có thể tìm thấy mô tả chặt chẽ cú pháp ngôn ngữ PADRE trong [1] hay mô tả mở rộng của nó trong [6].

Để khắc phục những phiền phức khi mô tả, biến đổi và thể hiện các lớp mạng Petri khác nhau bằng các ngôn ngữ lập trình cấp cao hiện có, chúng tôi đã thiết kế ngôn ngữ PADRE được trang bị hệ thống kiểu dữ liệu và tập các chỉ thị và các câu lệnh sau đây:

2.1. Hệ thống các kiểu dữ liệu

Ngoài các kiểu dữ liệu sơ cấp chuẩn như CHAR, BOOLEAN, INTEGER, và miền con các số nguyên, ngôn ngữ PADRE còn có kiểu *xây mạng* với cú pháp được thể hiện bằng công thức chuẩn EBNF ([7]) sau đây:

net – id = « NET OF »
[const – def « ; »]
[type – def « ; »]
element – decl « ; »
connection – clause
« END »

ở đây element – decl = « ELEMENT » elem – item { « ; » elem – item }
elem – item = elem – id { « , » elem – id } « : » elem – struct
elem – struc = « TRANSITION » | « SIMPLE PACE » | « COLORED PLACE OF »
int – subrange – id
connection – clause = « CONNECTION » trans – connect [inteface]
{« | » trans – connect [interface]}

Trans – connect có cú pháp đơn giản như sau:
Trans – connect = trans – id « [»
[in – clause « ; »]
[guard – clause « ; »]
[out – clause]
«] »

ở đây với mỗi tên cái chuyển trans–id:

– in–clause đặc tả tất cả những in–places và những tiền điều kiện kèm theo của nó.

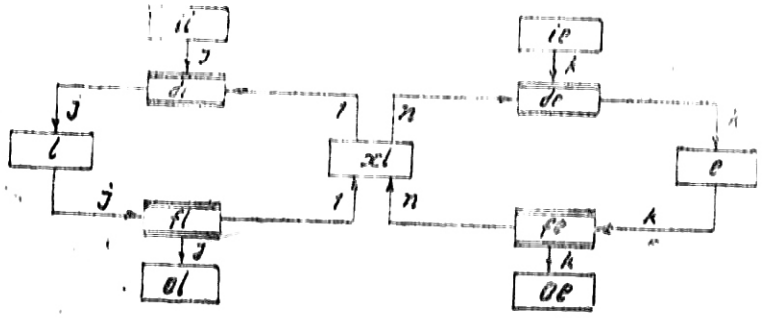
– guard – clause chỉ ra những ràng buộc bổ sung áp lên các biến có trong tiền điều kiện hoặc mặt tiếp giáp đồng bộ của nó với bên ngoài.

– out–clause đặc tả tất cả những out–places cùng với những hậu điều kiện kèm theo.

– interface đặc tả mặt tiếp giáp với bên ngoài.

Thay cho việc đi sâu vào những chi tiết cú pháp cụ thể đã được mô tả đầy đủ trong [6], chúng ta sẽ xem xét một ví dụ mô tả mạng Petri (đã chỉ ra trong [1]).

Ví dụ 1: Trên hình vẽ 1 mạng biểu diễn lời giải, trong đó không có cung *ức chế* của bài toán đọc ghi cổ điển ([8]). Sự liên kết trung thập kéo theo 3 ràng buộc quen biết là (i) các thao tác ghi loại trừ lẫn nhau, (ii) *cho phép* các thao tác đọc tương tranh, (iii) các thao tác đọc và ghi loại trừ lẫn nhau.



Hình 1

ở đây il (tương ứng ie) là vị trí chứa tất cả các quá trình đợi để đọc (tương ứng ghi),
l (tương ứng e) là vị trí chứa tất cả các quá trình đọc (tương ứng ghi),
ol (tương ứng oe) là vị trí chứa tất cả các quá trình đã đọc (tương ứng ghi) xong,
xl là simple place,

di (tương ứng Fi) là cái chuyển biểu diễn điểm đầu
(tương ứng điểm cuối) của quá trình đọc,

de (tương ứng fe) là cái chuyển biểu diễn điểm đầu
(tương ứng điểm cuối) của quá trình ghi.

Rõ ràng là chúng ta đã có một mạng màu, nó có thể mô tả bằng ngôn ngữ PADRE như sau

```
const n = 13;
type np = 1.. n;
rwnet = NET OF
        ELEMENT il, l ol, ie, e, oe: COLORED PLACE
                                     OF np;
                xl: SIMPLE PLACE;
                di, fi, de, fe: TRANSITION;
        CONNECTION
di[IN xl: (xl >= 1), il(j: np): (il [j] >= 1);
   OUT l: (1 TO l [j])
fi [ IN l(j: np): (l [j] >= 1);
   OUT xl: (1 TO xl), ol: (1 TO ol [ j ])] |
de [ IN xl: (xl = n), ie(k: np): (ie [k] > 1);
   OUT e: (1 TO e[k]) |
fe [IN e (k: np): (e[k]) = 1);
   OUT xl: (n TO xl), oe: (1 TO oe [k])]
   END;
```

Trên cơ sở *toán tử xây mạng*, ta có các thao tác sau đây trên mạng:

– Các thao tác bên trong mạng: các thao tác này cho phép thêm vào hay loại bỏ các phần tử của mạng.

– Các thao tác trên các mạng: như hợp 2 mạng bằng cách hợp nhất các phần tử có tên giống nhau, hợp các mạng bằng cách hợp nhất các phần tử có tên tương minh hoặc hợp các mạng bằng cách hợp 2 tập vị trí.

2.2. Tập các câu lệnh và chỉ thị

Tập này bao gồm các câu lệnh và các chỉ thị sau đây:

2.2.1. Các câu lệnh và chỉ thị truyền thống

– Câu lệnh gán, được định nghĩa như sau:

Biến « : = » biểu thức;

– Câu lệnh chọn có điều kiện được mô tả nhờ câu lệnh if

« IF »

guarded – command {« | » guarded – command}

« FI »

guarded – command = guard « : » statement – sequence;

guard = boolean – expression;

statement – sequence = statement {« ; » statement}

Trong câu lệnh IF, mỗi guarded – command sẽ được kiểm tra. Nếu guard tương ứng của nó đúng thì dãy câu lệnh statement – sequence sẽ được thực hiện.

– Câu lệnh LOOP không tiền định

« LOOP » guaaded command {« | » guarded command}

« POOL »

Trong câu lệnh LOOP, các câu lệnh guarded – command được kiểm tra ngẫu nhiên Nếu guard tương ứng đúng thì dãy câu lệnh statement – sequence đã được thực hiện. Câu lệnh LOOP chấm dứt thực hiện khi tất cả các guards đều sai. Câu lệnh IF và LOOP lấy ra từ [9] với đôi chút sửa đổi về ngữ nghĩa.

– Chỉ thị vào/ra: Tất cả các chỉ thị vào/ra của ngôn ngữ Pascal đều được giữ nguyên trong ngôn ngữ PADRE.

2.2.2. *Những chỉ thị riêng với mạng*

— Phát sinh một trạng thái tức thời của mạng. Chỉ thị « CREATE » «(» net — var — id « ; » net — tupe — id »)» phát sinh một trạng thái tức thời của mạng có tên net — var — id với kiểu được đặc tả trong net — type — id.

— Hủy bỏ một trạng thái tức thời của mạng. Chỉ thị « DELETE » «(» net — var — id«)» sẽ hủy bỏ trạng thái tức thời có tên net — var — id và trả lại phần bộ nhớ bị chiếm.

— Thể hiện ngẫu nhiên một mạng. Chỉ thị « RANDOMEXEC » «(» net — var — id «)» sẽ kích hoạt quá trình thể hiện ngẫu nhiên một trạng thái mạng tức thời net — var — id. Quá trình đó kết thúc khi tất cả các cái chuyển đã được chuẩn bị để cháy. Do chọn ngẫu nhiên, nên một cái chuyển có thể được chọn nhiều lần.

— Thể hiện một mạng có ưu tiên. Chỉ thị « PRIORITYEXEC » «(» net — vac — id «)» sẽ kích hoạt quá trình thể hiện mạng trong đó các cái chuyển được chọn để cháy tùy thuộc vào độ ưu tiên tương đối của chúng.

2.2.3. **Các chỉ thị cho phép xử lý và thể hiện các mạng theo chế độ hội thoại**

— Chỉ thị

« STEPBYSTEP » «(» net — var — id«)»

cho phép truy nhập đến trạng thái tức thời net — var — id và tạo ra những khả năng hội thoại sau đây:

+ Hiện các đánh dấu mạng và ma trận liên kết trên màn hình.

+ liệt kê các vị trí và/hay hiện danh sách các cái chuyển của mạng lên màn hình.

+ hiện những cái chuyển có thể cháy tại thời điểm đang xét lên màn hình.

+ truy nhập đến một vị trí hay một cái chuyển nào đó để thay đổi và / hoặc kiểm tra các thuộc tính của nó.

— Chỉ thị

« NETSWORK » «(» net-var-id1 « ; » « net-var-id2 «)» tạo ra những thao tác trên hai trạng thái tức thời được đặc tả nhờ tham số như sau:

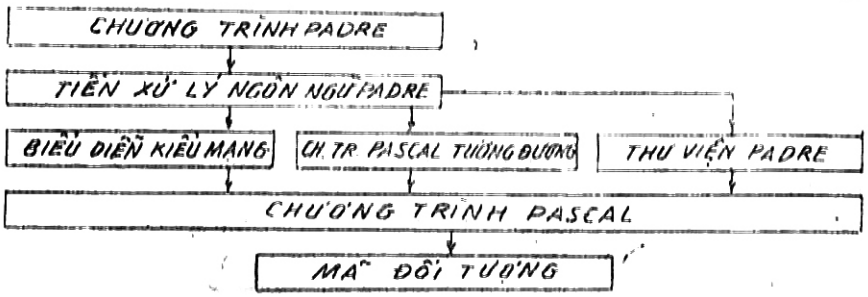
+ hợp nhất hai vị trí hay hai cái chuyển,

+ hợp nhất những phần tử có tên giống nhau,

+ hợp nhất hai tập vị trí ([10]).

3. CÀI ĐẶT NGÔN NGỮ PADRE

Quá trình cài đặt có thể mô tả nhờ sơ đồ sau



Ta có thể phân nhỏ quá trình cài đặt thành 2 bước, ở bước đầu mô tả quá trình dịch phần khai báo bằng ngôn ngữ PADRE và ở bước hai mô tả quá trình dịch phần lệnh.

3.1 Dịch *phần khai báo*

Dịch những kiểu dữ liệu chuẩn nguyên thủy như char, boolean, integer, miền con số nguyên không có khó khăn gì lớn lắm là do sự tương tự về cú pháp giữa Pascal và Padre.

Nhưng dịch *kiểu dữ liệu xây mạng* lại đòi hỏi phải phân tích sâu sắc những mô tả kiểu mạng để phát sinh động những biểu diễn trong của mạng. Để có thể sử dụng được những mạng lớn, tất cả những biểu diễn trong được tạo ra sẽ lưu trữ ở bộ nhớ ngoài.

Với mỗi mô tả kiểu mạng, bộ dịch sẽ tạo sinh biểu diễn trong tương ứng, một đồ thị ít nhiều phức tạp trong đó tất cả những thông tin thích hợp để sử dụng về sau chẳng hạn như các tiền điều kiện, guard, hậu điều kiện, các mặt tiếp giáp với bên ngoài, được thay đổi một cách phù hợp và lưu giữ lại nguyên vẹn.

Với mỗi thao tác bên trong mạng hay giữa các mạng với nhau, trong thư viện của ngôn ngữ PADRE có một thuật toán tương ứng đã được thử nghiệm tốt. Do vậy thư viện này phải gồm ít nhất một họ các thuật toán thao tác trên đồ thị như :

— Thuật toán duyệt đồ thị chẳng hạn để phát sinh những tiền điều kiện hay hậu điều kiện cho mỗi cái chuyển.

— Thuật toán sao chép đồ thị để phát sinh một thể hiện tức thời của mạng.

— Thuật toán hợp nhất đồ thị để thể hiện các phép kết hợp các mạng lại với nhau.

— Thuật toán thu dọn bộ nhớ khi có một trạng thái tức thời của một mạng nào đó bị hủy bỏ.

3.2 Dịch phần lệnh

Nói chung quá trình dịch phần lệnh không có gì khó khăn lắm, chính là do có sự tương đương trực tiếp về ngữ nghĩa giữa các cấu trúc điều khiển trong các ngôn ngữ Pascal và Padre. Có lẽ chỉ trừ các câu lệnh guarded-command và các chỉ thị thể hiện và xử lý mạng.

Dịch câu lệnh guarded-command

— Câu lệnh IF. Câu lệnh IF viết bằng ngôn ngữ PADRE

```
IF guard – 1 : statement – sequence 1 |
   guard – 2 : satatement – sequence 2 |
   ........
   guard – k : statement – sequence – k |
FI
```

được dịch thành dãy các câu lệnh Pascal sau :

```
if T(guard – 1) then begin T(statement – sequence – 1) end ;
if T(guard – 2) then begin T(statement – sequence – 2) end .
   .......
if T(guard – k) then begin T(statement – sequence – k) end ;
```

ở đây T(A) là kết quả dịch của A

— Câu lệnh LOOP

Câu lệnh LOOP trong ngôn ngữ PADRE

```
LOOP guard – 1 : statement – sequece – 1 |
        .......
     guard – k : statement – sequence – k |
POOL
```

được dịch thành dãy các câu lệnh PASCAL sau ;

```
j : = 0 ;
repeat case j of
     0 : begin end ;
     1 : if (guard – 1) then begin T(statement – sequence 1) end ;
     2 : if T(guard – 2) then begin T(statement – sequence – 2 end ;
     k : if T(guard – k) then begin T(statement – sequence – k) end ;
   end {case} ;
   j : = alea (generator, k) ;
until not (T(guard – 1) or... or T(guard – k)).
```

Trong sơ đồ dịch này, cơ chế không tiền định trong câu lệnh LOOP được thể hiện thông qua chọn ngẫu nhiên để thực hiện các câu lệnh guarded – command. Khi sử dụng kỹ thuật phân tích cú pháp xem trước 1 ký hiệu, do số các câu lệnh không biết được trước khi vào câu lệnh, nên để có thể áp dụng được nó trong quá trình dịch, chúng tôi đã cho vào một câu lệnh câm :

O : *begin end* :

Dịch các *chỉ thị xử lý* và thể hiện mạng.

Quá trình dịch các chỉ thị về trạng thái tức thời và hủy bỏ các mạng rất dễ dàng vì mỗi chỉ thị tương ứng với một thuật toán trong thư viện, chẳng hạn :

– chỉ thị phát sinh trạng thái tức thời mạng sẽ kích hoạt thuật toán sao chép đồ thị,

– chỉ thị loại bỏ mạng sẽ kích hoạt thuật toán thu dọn bộ nhớ.

Mặt khác, quá trình dịch các cơ cấu thể hiện mạng khác nhau đòi hỏi 2 thủ tục cơ bản. Thủ tục đầu tiên thể hiện sự cháy của cái chuyển t được mô tả bởi :

firing – a – transition – t ;

i) : { nếu t không có một vị trí in–place nào thì dãy câu lệnh Pascal sau đây sẽ được phát sinh :}

« IF » [guard – of – t « AND »] precondition – of – t « THEN »

« BEGIN » [interface–activation « 3 »]
mark–consuming–sequence » ; »
mark–adding–sequence » ; »

« END »

ii) : { nếu t có ít nhất một vị trí in–place được tô màu thì dãy câu lệnh sau đây được phát sinh trong đó các tham số x,.., w là màu của vị trí in–place của t.
Khi đó thuật toán thể hiện sẽ thực hiện tìm kiếm tất cả các khả năng để kiểm tra giá trị chân lý của tiền điều kiện và guard của t }

```
xt;–alea (generator, max–of–type–of–x);
x: = xt;
. . . . .
ωt: = alea (generator, max–of–type–of–w);
ω: = ωt;
repeat x: = x mod max–of–type–of–x+1;
  . . . . .
repeat ω: = ω mod max–of–type–of–ω+1;
if [guard–of–t «AND»] precondition–of–t
then begin done: = true;
           [intéface–activation;]
            mark–consuming–sequence;
            mark–adding–sequence;
     end
until (ω = ωt) or done;
until (x = xt) or done;
```

Thủ tục thứ hai thể hiện một danh sách các cái chuyển mà p trỏ tới :

```
{Thể hiện danh sách p}
While p <> nil do
begin t: = transition–pointed–by–p;
      firing–a–transition–t;
      p: = transition–following–p;
end;
```

Với hai thủ tục này, các chỉ thị khác dễ dàng dịch sang ngôn ngữ PASCAL, chẳng hạn

– Chỉ thị RANDOMEXEC được dịch thành

```
R: =;
repeat t: = choise (T);
       firing–of–t;
until R = T;
```

Ở đây T là tập các cái chuyển trong mạng, R là tập hiện thời những cái chuyển được chọn ngẫu nhiên, và chọn một Thuật toán chọn ngẫu nhiên nào đó

—Chỉ thị PRIORITYEXEC được dịch thành:

Phát sinh danh sách p các cái chuyển tùy thuộc vào độ ưu tiên của chúng;

Thể hiện danh sách p;

Việc dịch các chỉ thị STEPBYSTEP và NETSWOK chỉ cần đến các cơ chế đã chỉ ra ở trên.

4. KẾT LUẬN

Hai mục đích được đưa ra ở trên khi cài đặt ngôn ngữ PADRE nhờ kỹ thuật tiền xử lý đã đạt được:

—Tốc độ thực hiện: Bộ tiền xử lý đã được viết xong và hiện nay đang hoạt động trên hệ thống MULTICS. Việc cài đặt nó đòi hỏi năng suất trong 12 tháng người với khoảng 10.000 dòng lệnh PASCAL.

—Khả năng trao chuyển: Việc chuyển hệ thống sang hệ máy VAX/UNIX đang được tiến hành, quá trình đó dường như không có một chút khó khăn gì và hệ thống sắp tới sẽ hoạt động.

Văn bản chương trình hiện nay của bộ tiền xử lý có thể chia như sau:

—127 K bytes cho mã nguồn viết bằng PASCAL.

—138 K bytes cho thư viện của ngôn ngữ PADRE viết bằng PASCAL.

—232 K bytes cho mã đích sau khi liên kết.

Vì vậy kỹ thuật cài đặt này hoàn toàn phù hợp cho phần lớn máy vi tính hiện đang sử dụng.

Nhận ngày 15-2-1987

Tài liệu tham khảo

1. Van—Lu Nguyên, « PADRE: A Petri nets based parallel programming languge » in Proceedings of the 4th Hungarian comp. Science conference, Gyor, July 1985 Budapest. Eds: M. Arato, I. Katai, L. Varga.
2. Charles ANDRE, « Systèmes à evolutions parallèles: Modélisation par réseaux à capacités et Analyse par abstraction ». Thèse d' Etat, Université de Nice, Février 1981.
3. K. Jensen, « Coloured Petri nets and the invariant — method » in Theoretical Compute Science, N° 14, 1981.
4. H.J. Genrich, K. Lautenbach, « The analysis of distributed systems by means of predicate/ Transition nets » in Semantics of concurrent computation, Lecture notes in Computer Sciences, N° 70.
5. N. Wirth, « Data structures+Algorithms = Programs », Prentice—Hall, 1976.
6. Van—Lu Nguyen, « PADRE: un language d' experimentation de programmation parallète et distribuée base sur les Réseaux de Petri » Thèse d' E'tat, Université Paris VI, à paraître 1986.
7. N. Wirth, « What can we do about the unnecessary diversity of notation for syntactic definitions », in CACM, November, 1977.
8. P.J. Coutois et. al. « Conourrent control with readers and writers », CACM, october, 1971.
9. E. W. Dijkstra, « Guarded commands, nondeterminacy and formal derivation of programs », CACM, August, 1975.
10. V.E. Kotov, « An algebra for parallelism based on Petri nets », in Lecture notes in Computer Sciences, N°64, 1978.

ABSTRACT

ON THE PADRE'S IMPLEMENTATION BY PREPROCESSOR TECHNIQUE

Different steps and advantages of preprocessor technique are shown in this paper by way of a concrete implemenlation of Padre, a programming language, especially designed for describing, transforming and interpreting various classe of Petri nets. We also show that the technique is perfectly applicable on microcomputers.