Chứng minh sự đúng đắn

Khoa Khoa học máy tính

Phân tích thuật toán

- Kiểm tra sự đúng đắn
  - Chỉ ra rằng thuật toán cho kết quả như mong đợi sau một số bước thực hiện
- Đánh giá hiệu quả
  - Đánh giá nguồn tài nguyên thuật toán sử dụng của máy tính
    - Thời gian
    - Bộ nhớ

Kiểm tra tính đúng đắn

- Thực nghiệm
  - Kiểm thử (testing)
    - Thực thi thuật toán trên tập các dữ liệu vào và quan sát kết quả
- Lý thuyết
  - Chứng minh sự đúng đắn (correctness proof)
    - Chứng minh rằng thuật toán cho kết quả đúng với mọi dữ liệu vào

Ưu nhược điểm

	Thực nghiệm	**Lý thuyết**
Ưu điểm	-Đơn giản hơn -Dễ thực hiện	-Bảo đảm tính đúng đắn
Nhược điểm	-Không bảo đảm hoàn toàn tính đúng đắn	-Khó thực hiện -Không thể áp dụng cho các thuật toán phức tạp

Chứng minh sự đúng đắn

- Một vài khái niệm
  - Tiền điều kiện và hậu điều kiện
  - Trạng thái của thuật toán
  - Các xác nhận
  - Chú thích thuật toán

Tiền điều kiện và hậu điều kiện

- Tiền điều kiện (preconditions)
  - Các tính chất mà dữ liệu vào phải thoả mãn
- Hậu điều kiện (postconditions)
  - Các tính chất mà kết quả của thuật toán phải thoả mãn
- Ví dụ
  - Tìm giá trị *m* nhỏ nhất trong mảng không rỗng x[1..n]

    preconditions: $n \geq 1$

    postconditions: $m = \min(x[i] \mid 1 \leq i \leq n)$

Trạng thái của thuật toán

- Trạng thái của thuật toán
  - là tập các *giá trị tương ứng với tất cả các biến được sử dụng* trong thuật toán
- Trong quá trình thực thi trạng thái thuật toán thay đổi
- Thuật toán là đúng nếu cuối cùng trạng thái của nó thoả mãn hậu điều kiện

Các xác nhận

- Xác nhận (assertion)
  - là một câu lệnh mô tả các ràng buộc trên trạng thái của thuật toán

- Ví dụ

  {x > 0}
  x = x + y
  {x > y}

- Các xác nhận được sử dụng
  - Chứng minh sự đúng đắn
  - Chú thích thuật toán
  - Viết tài liệu

Chú thích thuật toán

- Sử dụng các xác nhận để chú thích thuật toán
- Ví dụ

```
min (x[1..n])
begin
    {n ≥ 1}
    m = x[1]
    {m = x[1]}
    for i from 2 to n do
    {2 ≤ i ≤ n}
        if (m > x[i]) then
            m = x[i]
        endif
        {m = min(x[1], x[2], …, x[i])}
    endfor
    return (m)
end
```

Chứng minh sự đứng đắn

- Chứng minh đúng đắn một phần (partial correctness)
  - Chứng minh rằng khi dữ liệu vào thoả mãn tiền điều kiện thì kết quả thuật toán sẽ thoả mãn hậu điều kiện
- Chứng minh đúng đắn toàn phần (total correctness)
  - Chứng minh rằng thuật toán đúng đắn một phần và thuật toán dừng
- Các bước trung gian trong chứng minh sự đúng đắn
  - Phân tích trạng thái của thuật toán
  - Phân tích sự ảnh hưởng của mỗi bước xử lý đến trạng thái của thuật toán

Chứng minh sự đứng đắn

- Ký hiệu
  - P - tiền điều kiện
  - Q - hậu điều kiện
  - A - thuật toán

  - A đúng đắn nếu với dữ liệu vào thoả mãn P thì A sẽ
    - cho kết quả thoả mãn Q
    - dừng sau một số bước xử lý hữu hạn
  - Ký hiệu

$$\{P\}\ A\ \{Q\}$$

Chứng minh sự đứng đắn

- Các bước cơ bản
  - Xác định các tiền điều kiện và hậu điều kiện
  - Chú thích thuật toán bằng cánh chèn thêm các xác nhận liên quan đến trạng thái của thuật toán sao cho
    - tiền điều kiện được thoả mãn
    - xác nhận cuối cùng phải bao hàm hậu điều kiện
  - Chứng minh rằng mỗi bước xử lý, thuật toán đi từ xác nhận trước xử lý đến xác nhận sau xử lý

Các quy tắc chứng minh sự đúng đắn

- Một số quy tắc cho các cấu trúc lệnh cơ bản
  - Lệnh tuần tự
  - Lệnh điều kiện/rẽ nhánh
  - Lệnh lặp

Quy tắc lệnh tuần tự

Dãy lệnh tuần tự A

$\{P_0\}$
I_1
$\{P_1\}$
...

$\{P_{i-1}\}$
I_k
$\{P_i\}$

...

$\{P_{n-1}\}$
I_n
$\{P_n\}$

Quy tắc

Nếu

$P \Rightarrow P_0$

$\{P_{k-1}\}\ I_k\ \{P_k\}$, $k=2..n$

$P_n \Rightarrow Q$

Thì

$\{P\}\ A\ \{Q\}$

Nghĩa là:

Nếu
- -tiền điều kiện P bao hàm xác nhận đầu tiên
- -mỗi câu lệnh bao hàm xác nhận tiếp theo
- -xác nhận cuối cùng bao hàm hậu điều kiện

Thì
- -dãy lệnh tuần tự A đúng

Quy tắc lệnh tuần tự

- Ví dụ
  - Hai biến x và y nhận hai giá trị tương ứng a và b. Hoán đổi giá trị hai biến x và y.
  - P = {x=a, y=b}
  - Q = {x=b, y=a}

```
{x=a, y=b, tmp chưa có giá trị}
tmp = x
{x=a, y=b, tmp=a}
x = y
{x=b, y=b, tmp=a}
y = tmp
{x=b, y=a, tmp=a}
```

Quy tắc lệnh điều kiện

Lệnh điều kiện A

$\{P_0\}$
<u>if</u> (c) <u>then</u>
    $\{c, P_0\}$
    I_1
    $\{c, P_1\}$
<u>else</u>
    $\{NOT\ c, P_0\}$
    I_2
    $\{NOT\ c, P_2\}$
<u>endif</u>

Quy tắc

Nếu

- $P \Rightarrow P_0$
- c có giá trị
- $c\ AND\ P_1 \Rightarrow Q$
- $NOT\ c\ AND\ P_2 \Rightarrow Q$

Thì

$\{P\}\ A\ \{Q\}$

Nghĩa là:

Nếu
- -tiền điều kiện P bao hàm xác nhận đầu tiên
- -C có thể được định giá
- -cả hai nhánh đều bao hàm hậu điều kiện

Thì
- -lệnh điều kiện A đúng

Quy tắc lệnh điều kiện

- Ví dụ
  - Tìm giá trị nhỏ nhất của a và b với a ≠ b

    preconditions: a ≠ b

    postconditions: m = min(a,b)

```
Lệnh A

{a ≠ b}
if (a < b) then
    {a < b}
    m = a
    {a < b, m = a}
else
    {a > b}
    m = b
    {a > b, m = b}
endif
```

{a < b, m = a} bao hàm m = min(a,b)

và

{a > b, m = b} bao hàm m = min(a,b)

Vậy {preconditions} A {postconditions}

Quy tắc lệnh lặp

- Một lệnh vòng lặp là đúng khi
  1. Nếu nó dừng, nó thoả mãn hậu điều kiện
  2. Nó dừng sau một số bước hữu hạn

- Nếu chỉ tính chất 1 đúng thì chỉ là *đúng đắn một phần*

- Đúng đắn một phần được chứng minh bởi quy nạp toán học hoặc *bất biến vòng lặp*

- Đúng đắn toàn phần cần chứng minh thêm thuật toán *dừng*

Bất biến vòng lặp

Lệnh lặp A

```
P ⇒ {I}
while (c) do
    {c, I}
    m = a
    {I}
endwhile
{NOT c, I} ⇒ Q
```

Định nghĩa

Một *bất biến vòng lặp* I là một *xác nhận* thoả mãn:

1. Bất biến vòng lặp đúng khi bắt đầu vòng lặp
2. Trong quá trình lặp (tức là điều kiện c đúng) thì bất biến vòng lặp I luôn đúng
3. Khi thoát khỏi vòng lặp (tức là điều kiện c sai) thì bất biến vòng lặp I phải bao hàm hậu điều kiện

Khi xác định được bất biến vòng lặp, nghĩa là đã chứng minh được thuật toán đúng đắn một phần

Bất biến vòng lặp

- Ví dụ
  - Tìm giá trị *m* nhỏ nhất trong mảng không rỗng x[1..n]

    preconditions: n ≥ 1

    postconditions: m = min(x[i] | 1 ≤ i ≤ n)

```
min (x[1..n])
begin
    m = x[1]
    for i from 2 to n do
        if (m > x[i]) then
            m = x[i]
        endif
    endfor
    return (m)
end
```

→

```
min (x[1..n])
begin
    i = 1, m = x[i]
    while (i < n) do
        i = i + 1
        if (m > x[i]) then
            m = x[i]
        endif
    endwhile
    return (m)
end
```

Bất biến vòng lặp

- Ví dụ

  P: $n \geq 1$

  Q: $m = \min(x[i] \mid 1 \leq i \leq n)$

```
min (x[1..n])
begin
    i = 1, m = x[i]
    {m = min(x[j], j=1..i)}
    while (i < n) do
        {i<n, m = min(x[j], j=1..i)}
        i = i + 1
        if (m > x[i]) then
            m = x[i]
        endif
        {m = min(x[j], j=1..i)}
    endwhile
    {i=n, m = min(x[j], j=1..i)}
    return (m)
end
```

Bất biến vòng lặp:

I = {m = min(x[j], j=1..i)}

Bởi vì:

- nếu i=1, m=x[1] thì I đúng
- nếu i<n, sau khi thực thi thân vòng lặp, I vẫn đúng
- nếu i=n, m=min(x[j], j=1..n) chính là hậu điều kiện

Hàm dừng

- Để chứng minh vòng lặp dừng sau một số bước lặp hữu hạn, chỉ cần xác định hàm dừng (termination function)

- Định nghĩa
  - Hàm T: N→N được gọi là hàm dừng nếu nó thoả mãn:

    1. T luôn giảm

    2. Nếu điều kiện *c* đúng thì T(p)>0, nếu T(p)=0 thì điều kiện *c* sai

- Nhận xét
  - T phụ thuộc biến đếm của vòng lặp p
    - Sau lần lặp thư nhất p = 1, sau lần lặp thứ hai p = 2, …
  - T sẽ bằng 0 vì nó luôn giảm
  - Khi T bằng 0 thì điều kiện c sai nên vòng lặp dừng

Hàm dừng

- Ví dụ

```
min (x[1..n])
begin
    i = 1, m = x[i]
    while (i < n) do
        i = i + 1
        {i_p = i_{p-1} + 1}
        if (m > x[i]) then
            m = x[i]
        endif
    endwhile
    return (m)
end
```

Hàm dừng: $T(p) = n - i_p$

Bởi vì:

$T(p) = n - i_p = n - i_{p-1} - 1$
$= T(p-1) - 1$

Vậy $T(p) < T(p-1)$

Nghĩa là hàm T luôn giảm (p tăng dần)

Nếu điều kiện vòng lặp đúng,
thì $i_p < n$, vậy $T(p) > 0$
Nếu $T(p) = 0$, thì $n - i_p = 0$
Khi đó điều kiện vòng lặp sai

Ví dụ

- Tìm chỉ số của phần tử (1)
  - Cho mảng a[1..n] có chứa phần tử x. Tìm chỉ số i nhỏ nhất sao cho a[i] = x.
  - preconditions: $n \geq 1$, $\exists i \in [1..n]$: a[i]=x
  - postconditions: $\exists i \in [1..n]$: a[i]=x, $\forall k \in [1..i-1]$:a[k]$\neq$x
  - Chứng minh thuật toán sau là đúng

```
timphantu (a[1..n], x)
begin
    i = 1
    while (a[i] ≠ x) do
        i = i + 1
    endwhile
    return (i)
end
```

Ví dụ

- Tìm chỉ số của phần tử (2)
  - Xác định bất biến vòng lặp

```
timphantu (a[1..n], x)
begin
    i = 1
    {a[k] ≠ x, k=1..i-1}
    while (a[i] ≠ x) do
        {a[i] ≠ x, a[k] ≠ x, k=1..i-1}
        i = i + 1
        {a[k] ≠ x, k=1..i-1}
    endwhile
    {a[i] = x, a[k] ≠ x, k=1..i-1}
    return (i)
end
```

Bất biến vòng lặp:

$I = \{a[k] \neq x, k=1..i-1\}$

Chứng minh quy nạp:

-i=1, thì k=1..0 nên I đúng

-Giả sử I đúng sau bước lặp i, nghĩa là $a[k] \neq x$, k=1..i-1

-Ở bước lặp i+1:

-nếu $a[i+1] \neq x$ thì $a[k] \neq x$, k=1..i, nghĩa là I đúng

-nếu $a[i+1]=x$ thì ta có hậu điều kiện Q

Ví dụ

- Tìm chỉ số của phần tử (3)
  - Xác định hàm dừng

```
timphantu (a[1..n], x)
begin
    i = 1
    while (a[i] ≠ x) do
        i = i + 1
        {i_p=i_p-1+1}
    endwhile
    return (i)
end
```

Gọi k là chỉ số nhỏ nhất mà a[k]=x

Hàm dừng:

$T = k - i_p$

Thật vậy:

$T(p) = k - i_p = k - i_{p-1} - 1$

$= T(p-1) - 1$

Vậy $T(p) < T(p-1)$

Nghĩa là hàm T luôn giảm (p tăng dần)

Nếu điều kiện $a[i_p] \neq x$ đúng, thì $k > i_p$, vậy $T(p) > 0$

Nếu $T(p) = 0$, thì $k = i_p$, khi đó điều kiện $a[i_p] \neq x$ sai

Nhận xét

- Dễ dàng đối với các lệnh tuần tự và lệnh điều kiện
- Khó khăn đối với lệnh lặp
- Xác định bất biến vòng lặp nói chung là rất phức tạp
- Chứng minh sự đúng đắn đòi hỏi nhiều thời gian và công sức
- Không thể áp dụng đối với các thuật toán phức tạp

Bài tập

1. Viết thuật toán tính n! và chứng minh thuật toán đúng đắn.

2. Thuật toán sau làm gì ? Chứng minh câu trả lời.

```
begin
    m = n
    k = 0
    b[0] = m MOD 2
    m = m DIV 2
    while (m ≠ 0) do
        k = k + 1
        b[k] = m MOD 2
        m = m DIV 2
    endwhile
end
```

Bài tập

3. Thuật toán sau làm gì ? Chứng minh câu trả lời.

```
// cho b[1..k], ∀i: b[i] = 0 hoặc b[i] = 1
begin
    i = k
    n = b[i]
    while (i > 0) do
        i = i – 1
        n = 2*n + b[i]
    endwhile
end
```

4. Viết thuật toán tính giá trị trung bình cộng các phần tử của mảng a[1..n]. Chứng minh thuật toán đúng đắn.