PHENIKAA UNIVERSITY
Realizing your potential

CHƯƠNG TRÌNH DỊCH

Chương 4. Phân tích cú pháp
Thuật toán Earley

TS. Phạm Văn Cảnh
Khoa Công nghệ thông tin

Email: canh.phamvan@phenikaa-uni.edu.vn

Nội dung

PHENIKAA UNIVERSITY
Realizing your potential

1. Giới thiệu
2. Ý tưởng cơ bản
3. Mã minh họa
4. Ví dụ
5. Đánh giá thuật toán
6. Bài tập

1. Giới thiệu

- Thuật toán Earley được giới thiệu năm 1968 bởi Jay Earley (nhà khoa học máy tính và tâm lý học, người Mỹ)
  - Công trình về phân tích văn phạm được đánh giá là một trong 25 bài báo xuất sắc nhất của tạp chí "Communications of the A.C.M" trong 1/4 thế kỷ
  - Earley nổi tiếng hơn trong ngành tâm lý học lâm sàng, chuyên về trị liệu nhóm, tác giả của Pattern System

2. Ý tưởng cơ bản

- Thuật toán Earley cụ thể hóa một automat tuyến tính không quay lui (đi thẳng, từ trái qua phải)
- Trạng thái của automat: tập hợp các bộ quan sát, một bộ quan sát thực chất là một biến ghi nhận quá trình diễn tiến của việc phân tích văn phạm trong một tình huống cụ thể nào đó
- Khi nhận kí hiệu đầu vào, automat thực hiện việc cập nhật các bộ quan sát để xác định xem quá trình phân tích đã đến đâu
- Kết quả ở bước cuối cùng cho biết automat đoán nhận được những gì

2. Ý tưởng cơ bản

- Xét chuỗi vào $w = w_0 w_1 \dots w_{n-1}$
- Thuật toán sử dụng một automat xử lý từ trái sang phải (từ w_0 sang đến w_{n-1})
- Thuật toán sử dụng dấu chấm để ngăn giữa 2 phần của luật sinh trong quá trình áp dụng luật đó
- Nói cách khác, khi viết $A \rightarrow \alpha \bullet \beta$, ta hiểu phần α đã phân tích xong, còn phần β thì chưa
- Một bộ quan sát $[A \rightarrow \alpha \bullet \beta, i]$ có nghĩa đang xử lý luật $A \rightarrow \alpha \bullet \beta$ từ vị trí w_i trở đi

2. Ý tưởng cơ bản

- Khi automat xét đến kí hiệu w_m, có thể có nhiều phương án phân tích khác nhau, tất cả các phương án này đều được lưu lại để sử dụng trong các bước tính toán tiếp theo.
- Tập hợp $S(m)$: tập các bộ quan sát dừng tại vị trí m
- Như vậy, nếu $[A \rightarrow \alpha \bullet \beta, i]$ thuộc $S(m)$ có nghĩa là dãy $w_i w_{i+1} \dots w_{m-1}$ được đoán nhận bởi phần α trong luật sinh $A \rightarrow \alpha \bullet \beta$
- Thuật toán cần phải sinh mọi thành phần trong $S(m)$ trước khi chuyển sang kí hiệu w_{m+1}

2. Ý tưởng cơ bản

- Thuật toán sẽ tính lần lượt S(0), S(1),..., S(n)
- Để dễ dàng thực hiện thuật toán, thuật toán bổ sung luật $P \rightarrow S$ vào tập luật (gọi là start rule) và bổ sung bộ $[P \rightarrow \bullet\ S, 0]$ vào S(0)
- Khi nhận kí hiệu w_m, automat sẽ bổ sung vào S(m) các bộ quan sát phù hợp, quá trình tính S(m) dừng khi không còn bộ quan sát nào có thể thêm vào
- Sau khi tính xong S(n), nếu bộ $[P \rightarrow S\ \bullet, 0]$ thuộc S(n) có nghĩa là dãy $w_0 w_1 \ldots w_{n-1}$ có thể sinh bởi S

2. Ý tưởng: 3 lệnh cơ bản

- **Prediction (dự đoán):** với mọi bộ $[X \rightarrow \alpha \bullet Y\ \beta, j]$ thuộc S(k), ta tìm mọi luật sinh dạng $Y \rightarrow \gamma$ và bổ sung bộ $[Y \rightarrow \bullet\ \gamma, k]$ vào S(k)
- **Scanning (xét duyệt):** với kí hiệu kết thúc a = wk, tìm mọi bộ $[X \rightarrow \alpha \bullet a\ \beta, j]$ thuộc S(k), bổ sung vào S(k+1) bộ $[X \rightarrow \alpha\ a \bullet \beta, j]$
- **Completion (hoàn thành):** với mọi bộ $[X \rightarrow \gamma \bullet, j]$ thuộc S(k), tìm trong S(j) mọi bộ $[Y \rightarrow \alpha \bullet X\ \beta, i]$, bổ sung $[Y \rightarrow \alpha\ X \bullet \beta, i]$ vào S(k)

3. Mã minh họa: hàm chính

```
function EARLEY-PARSE(words, grammar)
    ENQUEUE((γ → •S, 0), chart[0])
    for i ← from 0 to LENGTH(words) do
        for each state in chart[i] do
            if INCOMPLETE?(state) then
                if NEXT-CAT(state) is a nonterminal then
                    PREDICTOR(state, i, grammar)
                else do
                    SCANNER(state, i)
            else do
                COMPLETER(state, i)
        end
    end
    return chart
```

3. Mã minh họa: 3 lệnh cơ bản

```
procedure PREDICTOR((A → α•B, i), j, grammar)
    for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar) do
        ADD-TO-SET((B → •γ, j), chart[j])
    end

procedure SCANNER((A → α•B, i), j)
    if B ⊂ PARTS-OF-SPEECH(word[j]) then
        ADD-TO-SET((B → word[j], i), chart[j + 1])
    end

procedure COMPLETER((B → γ•, j), k)
    for each (A → α•Bβ, i) in chart[j] do
        ADD-TO-SET((A → αB•β, i), chart[k])
    end
```

4. Ví dụ

Bộ luật:

$P \rightarrow S$ // start rule

$S \rightarrow S + M \mid M$

$M \rightarrow M * T \mid T$

$T \rightarrow 1 \mid 2 \mid 3 \mid 4$

Chuỗi w = 2 + 3 * 4

4. Ví dụ

S(0): • 2 + 3 * 4

(1)	P → • S	(0) # start rule
(2)	S → • S + M	(0) # predict từ (1)
(3)	S → • M	(0) # predict từ (1)
(4)	M → • M * T	(0) # predict từ (3)
(5)	M → • T	(0) # predict từ (3)
(6)	T → • number	(0) # predict từ (5)

S(1): 2 • + 3 * 4

(1)	T → number •	(0) # scan từ S(0)(6)
(2)	M → T •	(0) # complete từ (1) và S(0)(5)
(3)	M → M • * T	(0) # complete từ (2) và S(0)(4)
(4)	S → M •	(0) # complete từ (2) và S(0)(3)
(5)	S → S • + M	(0) # complete từ (4) và S(0)(2)
(6)	P → S •	(0) # complete từ (4) và S(0)(1)

4. Ví dụ

S(1): 2 • + 3 * 4

```
(1)  T → number •      (0)    # scan từ S(0)(6)
(2)  M → T •           (0)    # complete từ (1) và S(0)(5)
(3)  M → M • * T       (0)    # complete từ (2) và S(0)(4)
(4)  S → M •           (0)    # complete từ (2) và S(0)(3)
(5)  S → S • + M       (0)    # complete từ (4) và S(0)(2)
(6)  P → S •           (0)    # complete từ (4) và S(0)(1)
```

S(2): 2 + • 3 * 4

```
(1)  S → S + • M       (0)    # scan từ S(1)(5)
(2)  M → • M * T       (2)    # predict từ (1)
(3)  M → • T           (2)    # predict từ (1)
(4)  T → • number      (2)    # predict từ (3)
```

4. Ví dụ

S(2): 2 + • 3 * 4

(1)	S → S + • M	(0)	# scan từ S(1)(5)
(2)	M → • M * T	(2)	# predict từ (1)
(3)	M → • T	(2)	# predict từ (1)
(4)	T → • number	(2)	# predict từ (3)

S(3): 2 + 3 • * 4

(1)	T → number •	(2)	# scan từ S(2)(4)
(2)	M → T •	(2)	# complete từ (1) và S(2)(3)
(3)	M → M • * T	(2)	# complete từ (2) và S(2)(2)
(4)	S → S + M •	(0)	# complete từ (2) và S(2)(1)
(5)	S → S • + M	(0)	# complete từ (4) và S(0)(2)
(6)	P → S •	(0)	# complete từ (4) và S(0)(1)

4. Ví dụ

S(3): 2 + 3 • * 4

(1)	T → number •	(2)	# scan từ S(2)(4)
(2)	M → T •	(2)	# complete từ (1) và S(2)(3)
(3)	M → M • * T	(2)	# complete từ (2) và S(2)(2)
(4)	S → S + M •	(0)	# complete từ (2) và S(2)(1)
(5)	S → S • + M	(0)	# complete từ (4) và S(0)(2)
(6)	P → S •	(0)	# complete từ (4) và S(0)(1)

S(4): 2 + 3 * • 4

(1)	M → M * • T	(2)	# scan từ S(3)(3)
(2)	T → • number	(4)	# predict từ (1)

4. Ví dụ

S(4): 2 + 3 * • 4

(1)	M → M * • T	(2)	# scan từ S(3)(3)
(2)	T → • number	(4)	# predict từ (1)

S(5): 2 + 3 * 4 •

(1)	T → number •	(4)	# scan từ S(4)(2)
(2)	M → M * T •	(2)	# complete từ (1) và S(4)(1)
(3)	M → M • * T	(2)	# complete từ (2) và S(2)(2)
(4)	S → S + M •	(0)	# complete từ (2) và S(2)(1)
(5)	S → S • + M	(0)	# complete từ (4) và S(0)(2)
(6)	P → S •	(0)	# complete từ (4) và S(0)(1)

Bộ [P → S •, 0] thuộc S(5), như vậy có thể kết luận chuỗi w được suy dẫn từ P

5. Đánh giá thuật toán

- Nhiều phiên bản cài đặt sau này có sửa đổi chút ít so với thuật toán gốc (thuật toán được giới thiệu trong slide này cũng không phải thuật toán gốc)
- Là một sự kết hợp thông minh của 3 trường phái
  - Tiếp cận top-down (bước prediction)
  - Tiếp cận bottom-up (bước scanning và completion)
  - Quy hoạch động (lưu lại trạng thái để dùng lại)
- Không bị hạn chế văn phạm đầu vào
  - Do là top-down nên không bị hạn chế bởi suy dẫn rỗng
  - Dùng quy hoạch động không bị hạn chế bởi ký hiệu đệ quy (hoặc đệ quy trái)

5. Đánh giá thuật toán

- Làm việc trực tiếp với luật dạng CFG: không cần phải tách thành các luật chuẩn chomsky, vì vậy kích cỡ tập luật không quá lớn
- Trong tình huống tổng quát: có độ phức tạp tính toán $O(n^3)$ với n là độ dài chuỗi vào
- Nếu văn phạm không có nhập nhằng: độ phức tạp tính toán cỡ $O(n^2)$
- Nếu văn phạm đơn giản (dạng LL, LR,...): độ phức tạp cận tuyến tính ~O(n)
- Thực hiện đặc biệt tốt nếu văn phạm đệ quy trái

6. Bài tập