2.3. DỊCH HƯỚNG CÚ PHÁP

Ảnh có chứa hàng, Phông chữ, biểu đồ, màu trắng

Mô tả được tạo tự động

Hình 2.13: Một lá bổ sung được tạo ra cho một hành động ngữ nghĩa.

Các hành động được nhúng trong các cơ thể sản xuất chuyển đổi các biểu thức đó thành biểu diễn hậu tố, miễn là chúng ta thực hiện một duyệt qua cây từ trái sang phải và từ trên xuống dưới và thực hiện mỗi câu lệnh in khi chúng ta ghé thăm lá của nó.

Ảnh có chứa biểu đồ, hàng, màu trắng, Phông chữ

Mô tả được tạo tự động

Hình 2.14: Các hành động dịch 9-5+2 thành 95-2+

Ảnh có chứa văn bản, biên lai, Phông chữ, màu trắng

Mô tả được tạo tự động

Hình 2.15: Các hành động để dịch thành biểu diễn hậu tố

Gốc của Hình 2.14 đại diện cho sản xuất đầu tiên trong Hình 2.15. Trong một duyệt theo thứ tự hậu tử, chúng ta đầu tiên thực hiện tất cả các hành động trong cây con bên trái của gốc, cho toán hạng bên trái, cũng được đánh dấu là expr giống như gốc. Tiếp theo, chúng ta ghé thăm lá + nơi không có hành động nào. Chúng ta sau đó thực hiện các hành động trong cây con cho toán hạng bên phải term và, cuối cùng, hành động ngữ nghĩa {in('+')} ở nút bổ sung.

Vì các sản xuất cho term chỉ có một chữ số ở bên phải, số đó được in ra bởi các hành động cho các sản xuất. Không cần phải xuất ra gì cho sản xuất expr + term, và chỉ có toán tử cần phải được in ra.

Hành động cho hai sản xuất đầu tiên. Khi được thực thi trong một duyệt theo thứ tự hậu tử của cây phân tích cú pháp, các hành động trong Hình 2.14 in ra 95-2+.

Lưu ý rằng mặc dù các phương pháp trong Hình 2.10 và Hình 2.15 tạo ra cùng một bản dịch, nhưng chúng xây dựng nó theo cách khác nhau; Hình 2.10 gắn các chuỗi như các thuộc tính vào các nút trong cây phân tích cú pháp, trong khi phương pháp trong Hình 2.15 in ra bản dịch theo cách tăng dần, thông qua các hành động ngữ nghĩa.

Các hành động ngữ nghĩa trong cây phân tích cú pháp trong Hình 2.14 chuyển đổi biểu thức trung tố 9-5+2 thành 95-2+ bằng cách in ra mỗi ký tự trong 9-5+2 chính xác một lần, mà không sử dụng bất kỳ bộ nhớ nào cho bản dịch của các biểu thức con. Khi đầu ra được tạo ra tăng dần theo cách này, thứ tự mà các ký tự được in ra là quan trọng.

Việc triển khai của một phương pháp dịch phải đảm bảo rằng các hành động ngữ nghĩa được thực hiện theo thứ tự chúng sẽ xuất hiện trong một duyệt theo thứ tự hậu tử của cây phân tích cú pháp. Việc triển khai không cần thực sự xây dựng một cây phân tích cú pháp (thường thì không), miễn là đảm bảo rằng các hành động ngữ nghĩa được thực hiện như thể chúng ta xây dựng một cây phân tích cú pháp và sau đó thực hiện các hành động trong suốt một duyệt theo thứ tự hậu tử.

**2.3.6 Bài tập cho Phần 2.3:**

**Bài tập 2.3.1 :** Xây dựng một phương pháp dịch hướng cú pháp chuyển đổi các biểu thức toán học từ trung tố sang tiền tố trong đó một toán tử xuất hiện trước các toán hạng của nó; ví dụ, -xy là biểu diễn tiền tố cho x - y. Đưa ra cây phân tích được chú thích cho các đầu vào 9-5+2 và 9-5\*2.

**Bài tập 2.3.2 :** Xây dựng một phương pháp dịch hướng cú pháp chuyển đổi các biểu thức toán học từ dạng hậu tố sang dạng trung tố. Đưa ra cây phân tích được chú thích cho các đầu vào 95-2\* và 952\*-.

**Bài tập 2.3.3 :** Xây dựng một phương pháp dịch hướng cú pháp chuyển đổi các số nguyên thành số La Mã.

**Bài tập 2.3.4 :** Xây dựng một phương pháp dịch hướng cú pháp chuyển đổi số La Mã thành số nguyên.

**Bài tập 2.3.5 :** Xây dựng một phương pháp dịch hướng cú pháp chuyển đổi các biểu thức toán học hậu tố thành các biểu thức toán học trung tố tương đương.

**2.4 PHÂN TÍCH CÚ PHÁP**

Phân tích cú pháp là quá trình xác định cách một chuỗi các ký hiệu có thể được tạo ra bởi một ngữ pháp. Khi thảo luận về vấn đề này, việc nghĩ về một cây phân tích cú pháp đang được xây dựng là hữu ích, mặc dù một trình biên dịch có thể không xây dựng một trong thực tế. Tuy nhiên, một trình phân tích cú pháp phải có khả năng xây dựng cây trong nguyên tắc, nếu không, việc dịch không thể đảm bảo đúng.

Phần này giới thiệu một phương pháp phân tích cú pháp được gọi là "đệ quy hạ cấp," có thể được sử dụng cả để phân tích cú pháp và để thực thi các trình dịch hướng cú pháp. Một chương trình Java hoàn chỉnh, thực hiện phương pháp dịch của Hình 2.15, xuất hiện trong phần tiếp theo. Một lựa chọn khả thi khác là sử dụng một công cụ phần mềm để tạo ra một trình dịch trực tiếp từ một phương pháp dịch. Phần 4.9 mô tả một công cụ như vậy - Yacc; nó có thể thực hiện phương pháp dịch của Hình 2.15 mà không cần sửa đổi.

Đối với bất kỳ ngữ pháp không gian không gian cụ thể nào, có một trình phân tích cú pháp mà cần tối đa O(n3) thời gian để phân tích một chuỗi của n ký hiệu terminals. Nhưng thời gian lập phương thường quá đắt đỏ. May mắn thay, đối với các ngôn ngữ lập trình thực tế, chúng ta thường có thể thiết kế một ngữ pháp mà có thể được phân tích cú pháp nhanh chóng. Thuật toán thời gian tuyến tính đủ để phân tích cú pháp hầu hết các ngôn ngữ xuất hiện trong thực tế. Các trình phân tích ngôn ngữ lập trình hầu như luôn luôn thực hiện một quét duy nhất từ trái sang phải qua đầu vào, nhìn trước một ký hiệu một lần và xây dựng các phần của cây phân tích khi chúng tiến hành.

Hầu hết các phương pháp phân tích cú pháp thuộc vào một trong hai lớp gọi là phương pháp từ trên xuống và từ dưới lên. Các thuật ngữ này đề cập đến thứ tự mà các nút trong cây phân tích được xây dựng. Trong các trình phân tích từ trên xuống, việc xây dựng bắt đầu từ gốc và tiến tới các lá, trong khi trong các trình phân tích từ dưới lên, xây dựng bắt đầu từ các lá và tiến tới gốc. Sự phổ biến của các trình phân tích từ trên xuống là do việc có thể dễ dàng xây dựng các trình phân tích hiệu quả hơn bằng tay bằng cách sử dụng các phương pháp từ trên xuống. Tuy nhiên, phân tích từ dưới lên có thể xử lý một loạt lớn các ngữ pháp và các phương pháp dịch, do đó các công cụ phần mềm để tạo ra các trình phân tích trực tiếp từ các ngữ pháp thường sử dụng các phương pháp từ dưới lên.

**2.4.1 Phân tích từ trên xuống**

Chúng tôi giới thiệu phân tích từ trên xuống bằng cách xem xét một ngữ pháp phù hợp với lớp các phương pháp này. Sau này trong phần này, chúng tôi xem xét việc xây dựng các trình phân tích từ trên xuống nói chung. Ngữ pháp trong Hình 2.16 tạo ra một phần nhỏ của các câu lệnh trong ngôn ngữ C hoặc Java. Chúng tôi sử dụng các ký hiệu in đậm if và for cho các từ khóa "if" và "for", để nhấn mạnh rằng các chuỗi ký tự này được xem xét như các đơn vị, tức là như các ký hiệu terminals đơn lẻ. Hơn nữa, ký hiệu **expr** đại diện cho biểu thức; một ngữ pháp hoàn chỉnh hơn sẽ sử dụng một ký hiệu không phải terminals và có các sản xuất cho ký hiệu không phải terminals expr. Tương tự, **other** là một ký hiệu terminal đại diện cho các cấu trúc câu lệnh khác.

Xây dựng từ trên xuống của một cây phân tích như trong Hình 2.17, được thực hiện bằng cách bắt đầu với gốc, được đánh dấu bằng ký hiệu không phải terminals khởi đầu stmt, và lặp lại thực hiện hai bước sau đây.

1. Ở nút N, được đánh dấu bằng ký hiệu không phải terminals A, chọn một trong các sản xuất cho A và xây dựng các nút con ở N cho các ký hiệu trong cơ thể sản xuất.

2. Tìm nút tiếp theo mà tại đó một cây con sẽ được xây dựng, thường là ký hiệu không phải terminals chưa được mở rộng bên trái của cây.

Đối với một số ngữ pháp, các bước trên có thể được thực hiện trong một quét duy nhất từ trái sang phải của chuỗi đầu vào. Ký hiệu terminals hiện tại đang được quét trong

Ảnh có chứa văn bản, Phông chữ, biên lai, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 2.16: Một ngữ pháp cho một số câu lệnh trong C và Java

Ảnh có chứa hàng, bản phác thảo, biểu đồ, màu trắng

Mô tả được tạo tự động

Hình 2.17: Một cây phân tích dựa trên ngữ pháp trong Hình 2.16

Đầu vào thường được gọi là ký hiệu nhìn trước. Ban đầu, ký hiệu nhìn trước là ký tự terminal đầu tiên của chuỗi đầu vào, tức là ký tự ở bên trái nhất của chuỗi. Hình 2.18 minh họa việc xây dựng cây phân tích trong Hình 2.17 cho chuỗi đầu vào

**for ( ; expr ; expr ) other**

Ban đầu, ký hiệu nhìn trước là ký hiệu "for", và phần đã biết của cây phân tích bao gồm gốc, được đánh dấu bằng ký hiệu không phải terminals khởi đầu stmt trong Hình 2.18(a). Mục tiêu là xây dựng phần còn lại của cây phân tích sao cho chuỗi được tạo ra bởi cây phân tích phù hợp với chuỗi đầu vào.

Để có sự phù hợp xảy ra, ký hiệu không phải terminals stmt trong Hình 2.18(a) phải tạo ra một chuỗi bắt đầu bằng ký hiệu nhìn trước "**for**". Trong ngữ pháp của Hình 2.16, chỉ có một sản xuất cho stmt có thể tạo ra một chuỗi như vậy, vì vậy chúng ta chọn nó và xây dựng các nút con của gốc được đánh dấu bằng các ký hiệu trong cơ thể sản xuất. Sự mở rộng của cây phân tích được thể hiện trong Hình 2.18(b).

Mỗi trong ba ảnh chụp trong Hình 2.18 có các mũi tên đánh dấu ký hiệu nhìn trước trong đầu vào và nút trong cây phân tích đang được xem xét. Khi các nút con được xây dựng tại một nút, chúng ta tiếp tục xem xét nút con ở bên trái. Trong Hình 2.18(b), các nút con chỉ mới được xây dựng tại gốc, và nút con ở bên trái được đánh dấu bằng "**for**" đang được xem xét.

Khi nút trong cây phân tích đang được xem xét là một ký hiệu terminals, và ký hiệu terminals phù hợp với ký hiệu nhìn trước, thì chúng ta tiến vào cả cây phân tích và đầu vào. Ký hiệu terminals tiếp theo trong đầu vào trở thành ký hiệu nhìn trước mới, và nút con tiếp theo trong cây phân tích được xem xét. Trong Hình 2.18(c), mũi tên trong cây phân tích đã tiến tới nút con tiếp theo của gốc, và mũi tên

**Ảnh có chứa văn bản, hàng, biểu đồ, ảnh chụp màn hình

Mô tả được tạo tự động**

Hình 2.18: Phân tích từ trên xuống khi quét đầu vào từ trái sang phải

Trong đầu vào đã tiến tới terminal tiếp theo, đó là (. Một tiến bộ tiếp theo sẽ đưa mũi tên trong cây phân tích đến nút con được đánh dấu bằng ký hiệu không phải terminals optexpr và đưa mũi tên trong đầu vào đến terminal ;.

Tại nút không phải terminals được đánh dấu là optexpr, chúng ta lặp lại quá trình chọn một sản xuất cho một ký hiệu không phải terminals. Các sản xuất có t là cơ thể (còn gọi là "e-productions") yêu cầu xử lý đặc biệt. Trong lúc này, chúng ta sử dụng chúng như một giá trị mặc định khi không có sản xuất nào khác có thể được sử dụng; chúng ta sẽ quay lại vấn đề này trong Phần 2.4.3. Với ký hiệu không phải terminals optexpr và ký hiệu nhìn trước là ;, sản xuất "e" được sử dụng, vì ; không phù hợp với sản xuất duy nhất khác cho optexpr, mà có ký hiệu terminals **expr** là cơ thể.

Nói chung, việc chọn một sản xuất cho một ký hiệu không phải terminals có thể liên quan đến thử và sai; nghĩa là, chúng ta có thể phải thử một sản xuất và quay lại thử một sản xuất khác nếu sản xuất đầu tiên được xác định là không phù hợp. Một sản xuất là không phù hợp nếu, sau khi sử dụng sản xuất đó, chúng ta không thể hoàn thành cây để phù hợp với đầu vào.

**2.4.2 Phân Tích Dự Đoán**

Phân tích đệ quy hạ cấp là một phương pháp từ trên xuống của phân tích cú pháp trong đó một tập hợp các thủ tục đệ quy được sử dụng để xử lý đầu vào. Một thủ tục được liên kết với mỗi ký hiệu không phải terminals của một ngữ pháp. Ở đây, chúng ta xem xét một dạng đơn giản của phân tích đệ quy hạ cấp, được gọi là phân tích dự đoán, trong đó ký hiệu nhìn trước xác định một cách không đồng nhất luồng điều khiển thông qua thân thủ tục cho mỗi ký hiệu không phải terminals. Các cuộc gọi thủ tục trong quá trình phân tích của một chuỗi đầu vào ngầm định xác định một cây phân tích cho đầu vào và có thể được sử dụng để xây dựng một cây phân tích rõ ràng, nếu mong muốn.

Trình phân tích dự đoán trong Hình 2.19 bao gồm các thủ tục cho các ký hiệu không phải terminals stmt và optexpr của ngữ pháp trong Hình 2.16 và một thủ tục bổ sung match, được sử dụng để đơn giản hóa mã cho stmt và optexpr. Thủ tục match(t) so sánh đối số của nó t với ký hiệu nhìn trước và tiến tới ký hiệu tiếp theo của đầu vào nếu chúng phù hợp. Do đó, match thay đổi giá trị của biến nhìn trước, một biến toàn cục chứa ký hiệu terminal hiện đang quét của đầu vào.

Phân tích bắt đầu với một cuộc gọi của thủ tục cho ký hiệu không phải terminals khởi đầu stmt. Với cùng một đầu vào như trong Hình 2.18, nhìn trước ban đầu là ký hiệu terminals đầu tiên for. Thủ tục stmt thực thi mã tương ứng với sản xuất



Trong mã cho cơ thể sản xuất - tức là trường hợp for của thủ tục stmt - mỗi ký hiệu terminals được so sánh với ký hiệu nhìn trước, và mỗi ký hiệu không phải terminals dẫn đến một cuộc gọi của thủ tục tương ứng của nó, theo trình tự cuộc gọi sau đây:

Ảnh có chứa văn bản, Phông chữ, màu trắng, thư pháp

Mô tả được tạo tự động

Phân tích dự đoán phụ thuộc vào thông tin về các ký hiệu đầu tiên có thể được tạo ra bởi một cơ thể sản xuất. Cụ thể hơn, cho a là một chuỗi các ký hiệu ngữ pháp (terminals và/hoặc nonterminals). Chúng tôi định nghĩa FIRST(^) tb là tập hợp các ký hiệu terminals xuất hiện như các ký hiệu đầu tiên của một hoặc nhiều chuỗi terminals được tạo ra từ a. Nếu a là E hoặc có thể tạo ra t, thì t cũng thuộc FIRST(&). Các chi tiết về cách tính toán FIRST(^) được trình bày trong Phần 4.4.2. Ở đây, chúng tôi chỉ sử dụng lý lẽ tùy ý để suy ra các ký hiệu trong FIRST(^); thường, a sẽ bắt đầu bằng một ký hiệu terminal, do đó đó là ký hiệu duy nhất trong FIRST(^), hoặc a sẽ bắt đầu bằng một ký hiệu không phải terminals mà các cơ thể sản xuất của nó bắt đầu bằng terminals, trong trường hợp này các terminals này là các thành viên duy nhất của FIRST(^).

Ví dụ, đối với ngữ pháp trong Hình 2.16, các tính toán sau đây là đúng của FIRST.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động

Hình 2.19: Mã giả cho một trình phân tích dự đoán

Ảnh có chứa văn bản, Phông chữ, màu trắng, thư pháp

Mô tả được tạo tự động

Các tập hợp FIRST phải được xem xét nếu có hai sản xuất A -> a và A -> p. Bỏ qua các E-productions trong lúc này, phân tích dự đoán yêu cầu FIRST(A) và FIRST(B) phải không giao nhau. Ký hiệu nhìn trước sau đó có thể được sử dụng để quyết định sản xuất nào được sử dụng; nếu ký hiệu nhìn trước thuộc FIRST(A), thì a được sử dụng. Nếu không, nếu ký hiệu nhìn trước thuộc FIRST(B), thì p được sử dụng.

**2.4.3 Khi nào sử dụng các E-productions**

Trình phân tích dự đoán của chúng tôi sử dụng một E-production như là một mặc định khi không có sản xuất nào khác có thể được sử dụng. Với đầu vào trong Hình 2.18, sau khi các ký hiệu terminals for và ( được so khớp, ký hiệu nhìn trước là ;. Tại điểm này, thủ tục optexpr được gọi, và mã

ở trong Hình 2.19 cần được thực thi. Đây là một ví dụ về việc sử dụng sản xuất E như một giá trị mặc định khi không có sản xuất nào khác phù hợp để sử dụng.



Trong thân của nó được thực thi. Ký hiệu không phải terminals optexpr có hai sản xuất, với cơ thể là expr và E. Ký hiệu nhìn trước ";" không phù hợp với terminal expr, vì vậy sản xuất với cơ thể là expr không áp dụng được. Trong thực tế, thủ tục trả về mà không thay đổi ký hiệu nhìn trước hoặc làm bất cứ điều gì khác. Không làm gì tương ứng với áp dụng một sản xuất E.

Nói chung, xem xét một biến thể của các sản xuất trong Hình 2.16 nơi optexpr tạo ra một ký hiệu không phải terminals của biểu thức thay vì terminal expr:

Ảnh có chứa Phông chữ, văn bản, thuật in máy, màu trắng

Mô tả được tạo tự động

Do đó, optexpr có thể tạo ra một biểu thức bằng cách sử dụng ký hiệu không phải terminals expr hoặc nó tạo ra E. Trong quá trình phân tích optexpr, nếu ký hiệu nhìn trước không thuộc FIRST(expr), thì sản xuất E sẽ được sử dụng.

Để biết thêm về khi nào nên sử dụng các sản xuất E, xem phần thảo luận về các ngữ pháp LL(1) trong Phần 4.4.3.

**2.4.4 Thiết kế một Trình phân tích dự đoán**

Chúng ta có thể tổng quát hóa kỹ thuật được giới thiệu một cách không chính thức trong Phần 2.4.2, để áp dụng cho bất kỳ ngữ pháp nào có các tập hợp FIRST không giao nhau cho các cơ thể sản xuất thuộc về bất kỳ ký hiệu không phải terminals nào. Chúng ta cũng sẽ thấy rằng khi chúng ta có một kế hoạch dịch - tức là, một ngữ pháp với các hành động nhúng - thì có thể thực thi những hành động đó như một phần của các thủ tục được thiết kế cho trình phân tích.

Hãy nhớ rằng một trình phân tích dự đoán là một chương trình gồm một thủ tục cho mỗi ký hiệu không phải terminals. Thủ tục cho ký hiệu không phải terminals A thực hiện hai điều sau:

1. Nó quyết định sử dụng sản xuất A nào bằng cách kiểm tra ký hiệu nhìn trước. Sản xuất với cơ thể a (nơi a không phải E, chuỗi rỗng) được sử dụng nếu ký hiệu nhìn trước thuộc FIRST(A). Nếu có xung đột giữa hai cơ thể không rỗng cho bất kỳ ký hiệu nhìn trước nào, thì chúng ta không thể sử dụng phương pháp phân tích này trên ngữ pháp này. Ngoài ra, sản xuất E cho A, nếu tồn tại, được sử dụng nếu ký hiệu nhìn trước không thuộc tập hợp FIRST của bất kỳ cơ thể sản xuất nào khác cho A.

2. Sau đó, thủ tục mô phỏng cơ thể của sản xuất được chọn. Đó là, các ký hiệu của cơ thể được "thực thi" lần lượt, từ trái sang phải. Một ký hiệu không phải terminals được "thực thi" bằng cách gọi thủ tục cho ký hiệu không phải terminals đó, và một ký hiệu terminals phù hợp với ký hiệu nhìn trước được "thực thi" bằng cách đọc ký hiệu đầu vào tiếp theo. Nếu tại một thời điểm nào đó ký hiệu terminals trong cơ thể không phù hợp với ký hiệu nhìn trước, một lỗi cú pháp sẽ được báo cáo.

Hình 2.19 là kết quả của việc áp dụng các quy tắc này vào ngữ pháp trong Hình 2.16.

Như một kế hoạch dịch là được hình thành bằng cách mở rộng một ngữ pháp, một trình dịch hướng cú pháp có thể được hình thành bằng cách mở rộng một trình phân tích dự đoán. Một thuật toán cho mục đích này được đưa ra trong Phần 5.4. Việc xây dựng hạn chế sau đây đủ để hiện tại:

1. Xây dựng một trình phân tích dự đoán, bỏ qua các hành động trong các sản xuất.

2. Sao chép các hành động từ kế hoạch dịch vào trình phân tích. Nếu một hành động xuất hiện sau ký hiệu ngữ pháp X trong sản xuất p, thì nó được sao chép sau việc triển khai của X trong mã cho p. Nếu không, nếu nó xuất hiện ở đầu của sản xuất, thì nó được sao chép ngay trước mã cho thân sản xuất.

Chúng tôi sẽ xây dựng một trình dịch như vậy trong Phần 2.5.

**2.4.5 Đệ Quy Trái**

Có thể xảy ra trường hợp một trình phân tích đệ quy vô hạn. Một vấn đề nảy sinh với các sản xuất "đệ quy trái" như



Giả sử thủ tục cho expr quyết định áp dụng sản xuất này. Thân của sản xuất bắt đầu với expr nên thủ tục cho expr được gọi đệ qui. Vì ký hiệu nhìn trước thay đổi chỉ khi một ký hiệu terminal trong thân sản xuất được khớp, nên không có sự thay đổi nào đối với đầu vào xảy ra giữa các cuộc gọi đệ qui của expr. Kết quả là, cuộc gọi thứ hai cho expr thực hiện chính xác như cuộc gọi đầu tiên đã làm, điều này có nghĩa là một cuộc gọi thứ ba cho expr, và tiếp tục như vậy, mãi mãi.

Một sản xuất đệ quy trái có thể được loại bỏ bằng cách viết lại sản xuất gây ra vấn đề. Xem xét một ký hiệu không phải terminals A với hai sản xuất như sau:



trong đó a và b là các chuỗi của các ký hiệu terminals và nonterminals không bắt đầu bằng A. Ví dụ, trong



Ký hiệu không phải terminals A = expr, chuỗi a = + term và chuỗi b = term. Ký hiệu không phải terminals A và sản xuất của nó được gọi là đệ quy trái, vì sản xuất A -> Aa có chính A là ký hiệu trái nhất ở phía bên phải. Sự áp dụng lặp đi lặp lại của sản xuất này xây dựng một chuỗi các a ở bên phải của A, như trong Hình 2.20(a). Khi A cuối cùng được thay thế bằng P, chúng ta có một ,6 theo sau bởi một chuỗi của zero hoặc nhiều a.

Hiệu quả tương tự có thể được đạt được, như trong Hình 2.20(b), bằng cách viết lại các sản xuất cho A theo cách sau, sử dụng một ký hiệu không phải terminals mới R:

Trong một ngữ pháp đệ quy trái tổng quát, thay vì một sản xuất A -> Aa, ký hiệu không phải terminals A có thể suy ra Aa thông qua các sản xuất trung gian.

Ảnh có chứa biểu đồ, bản phác thảo, hàng, Bản vẽ kỹ thuật

Mô tả được tạo tự động

Hình 2.20: Cách sinh chuỗi bằng cách đệ quy trái và đệ quy phải

Ảnh có chứa Phông chữ, văn bản, thuật in máy, màu trắng

Mô tả được tạo tự động

Ký hiệu không phải terminals R và sản xuất của nó R -> cXR là đệ quy phải vì sản xuất này cho R có chính R là ký hiệu cuối cùng ở phía bên phải. Các sản xuất đệ quy phải dẫn đến cây mọc xuống về phía bên phải, như trong Hình 2.20(b).

Cây mọc xuống về phía bên phải làm cho việc dịch các biểu thức chứa các toán tử kết hợp trái, như trừ, trở nên khó khăn hơn. Tuy nhiên, trong Phần 2.5.2, chúng tôi sẽ thấy rằng việc dịch chính xác các biểu thức thành dạng postfix vẫn có thể đạt được thông qua việc thiết kế cẩn thận của kế hoạch dịch.

Trong Phần 4.3.3, chúng tôi sẽ xem xét các dạng tổng quát hơn của đệ quy trái và chỉ ra cách loại bỏ toàn bộ đệ quy trái từ một ngữ pháp.

**2.4.6 Bài tập cho Phần 2.4**

**Bài tập 2.4.1:** Xây dựng các trình phân tích đệ quy, bắt đầu với các ngữ pháp sau đây:

Ảnh có chứa văn bản, Phông chữ, màu trắng, thuật in máy

Mô tả được tạo tự động

**2.5 MỘT TRÌNH DỊCH CHO BIỂU THỨC ĐƠN GIẢN**

Sử dụng các kỹ thuật của ba phần trước đó, chúng ta bây giờ xây dựng một trình dịch hướng cú pháp, dưới dạng một chương trình Java hoạt động, chuyển đổi các biểu thức số học thành dạng postfix. Để giữ cho chương trình ban đầu có kích thước nhỏ và dễ quản lý, chúng tôi bắt đầu với các biểu thức gồm các chữ số được phân tách bởi dấu cộng và trừ nhị phân. Chúng tôi mở rộng chương trình trong Phần 2.6 để dịch các biểu thức bao gồm số và các toán tử khác. Điều này đáng giá để nghiên cứu.