1. **Chương trình dịch**

* là phần mềm hệ thống chuyển đổi đoạn văn viết trong ngôn ngữ A sang đoạn văn tương đương viết trong ngôn ngữ B

1. **Cấu trúc 1 chương trình dịch**

* Phân tích từ vựng
* Phân tích cú pháp
* Phân tích ngữ nghĩa
* Sinh mã trung gian
* Tối ưu mã trung gian
* Sinh mã đích

1. **Phân tích từ vựng**

* Nhiệm vụ:

+ Đọc dữ liệu đầu vào, loại bỏ các khối văn bản không cần thiết (dấu cách, dấu tab, các ghi chú, …)

+ Chia khối văn bản còn lại thành các từ vựng đồng thời xác định từ loại cho các từ vựng đó

* Từ khóa: if
* Ký hiệu: (, >=
* Tên riêng: a, max

1. **Phân tích cú pháp**

* Nhiệm vụ chính là sinh cây phân tích (hay cây cú pháp – syntax tree) cho dãy từ vựng
* Có rất nhiều phương pháp xây dựng một parser:

+ Sử dụng các kĩ thuật duyệt (top-down hoặc bottom-up)

+ Sử dụng kĩ thuật bảng phương án (automat đẩy xuống)

1. **Phân tích ngữ nghĩa**

* Dựa trên cây phân tích để thực hiện 2 việc chính:

+ Kiểm tra xem chương trình nguồn có các lỗi về ngữ nghĩa hay không

+Tổng hợp các thông tin phục vụ cho giai đoạn sinh mã

1. **Sinh mã trung gian**

* Các trình dịch sử dụng loại mã 3 địa chỉ (TAC – three-address code) hoặc mã SSA (static single assignment) làm mã trung gia

1. **Tối ưu mã trung gian**

* Tối ưu (optimization) mã trung gian tạo ra mã có tốc độ chạy nhanh hơn

1. **Sinh mã đích**

* Sinh mã đích (code generation) tạo ra mã đích (thường là dạng mã máy hoặc mã assembly) từ các mã TAC hoặc SSA đã được tối ưu

1. **Phân tích và tổng hợp**

* Font-end: Gồm 3 bước đầu tiên
* Back-end: Gồm 3 bước sau cùng

1. **Định nghĩa văn phạm**

* Văn phạm G là một hệ thống (Σ, Δ, P, S) trong đó:

+ Σ là tập hữu hạn các ký hiệu kết thúc (terminal)

+ Δ là tập hữu hạn các ký hiệu không kết thúc (nonterminal)

* Còn gọi là ký hiệu trung gian hay biến
* Σ ∩ Δ = ∅

+ S ∈ Δ gọi là ký hiệu khởi đầu (initial)

+ P là tập hữu hạn các cặp chuỗi (α, β) được gọi luật văn phạm (syntax rule) hay luật sinh

* Thường được viết là α → β
* Chuỗi α phải có ít nhất một ký hiệu không kết thúc

1. **Các lớp văn phạm**

* Lớp 0: unrestricted grammars (văn phạm tự do)

+ Không có ràng buộc gì về luật sinh

+ Tương đương với lớp các ngôn ngữ loại đệ quy đếm được (recursively enumerable languages)

+ Được đoán nhận bởi máy Turing

* Lớp 1: context-sensitive grammars (văn phạm cảm ngữ cảnh)

+ Các luật sinh α→β phải thỏa mãn điều kiện | α | ≤ | β |

+ Tương đương với lớp các ngôn ngữ cảm ngữ cảnh (context-sensitive languages)

+ Được đoán nhận bởi automat tuyến tính giới nội (LBA – linear bounded automaton)

* Lớp 2: context-free grammars (văn phạm phi ngữ cảnh)

+ Các luật sinh phải có dạng A → α trong đó A ∈ Δ

+ Tương đương với lớp các ngôn ngữ phi ngữ cảnh (context-free languages)

+ Đoán nhận bởi automat đẩy xuống (pushdown automaton)

* Lớp 3: regular grammars (văn phạm chính quy)

+ Các luật sinh chỉ có thể ở 1 trong 2 loại:

• A → a, A → Ba trong đó A, B ∈ Δ, a ∈ Σ

• A → a, A → aB trong đó A, B ∈ Δ, a ∈ Σ

+ Sinh ra các ngôn ngữ chính quy (regular languages)

+ Đoán nhận bởi automat hữu hạn (finite state automaton)

1. **Văn phạm chính quy (biểu thức chính quy)**

* Văn phạm chính quy giới hạn các luật có dạng: A → a | aB với điều kiện A, B ∈ Δ, a ∈ Σ
* Biểu thức chính quy sử dụng bộ kí pháp sau:

• Kí hiệu | có nghĩa là hoặc (or)

• Kí hiệu () để nhóm các thành phần

• Kí hiệu \* có nghĩa là lặp lại không, một hoặc nhiều lần

* + lặp lại một hoặc nhiều lần
* ? lặp lại không hoặc một lần
* Ví dụ sử dụng BTCQ để mô tả từ vựng:

+ Chữ cái = A | B | … | Z | a | b | … | z

+ Chữ số = 0 | 1 | … | 9

+ Tên = Chữ cái (Chữ cái | Chữ số) \*

+ Số nguyên = (Chữ số) +

+ Phép toán = + | - | \* | /

+ Phép so sánh = < | <= | > | >= | == | !=

* Cách ghi BTCQ đôi lúc còn mở rộng như sau:

+ [abcd]: tương đương với (a|b|c|d)

+ [a-z]: tương đương với (a|b|c|…|z)

+ [^a-c]: bất kì kí hiệu nào không phải a, b hoặc c

1. **Automat hữu hạn**

* Automat hữu hạn dùng để đoán nhận lớp ngôn ngữ chính quy
* Cấu trúc của automat hữu hạn gồm:

+ Bảng chuyển

+ Đầu đọc

+ Xâu vào

* Hoạt động của automat:

+ Bắt đầu từ trạng thái xuất phát

+ Đọc dữ liệu từ xâu vào

+ Quan sát bảng chuyển để biết sẽ chuyển sang trạng thái nào

+ Dừng khi kết thúc xâu vào và trả về trạng thái đoán nhận

* Phân loại:

+ Automat hữu hạn đơn định (deterministic finite automata – DFA)

• Với một kí hiệu đầu vào, chỉ có thể chuyển sang tối đa một trạng thái thái tiếp theo (hoặc dừng và báo lỗi)

• Không chấp nhận kí hiệu đầu vào là ε

+ Automat hữu hạn không đơn định (non-deterministic finite automata – NFA)

• Chấp nhận kí hiệu đầu vào là ε

• Với một kí hiệu đầu vào, có thể chuyển sang nhiều trạng thái tiếp theo

1. **Văn phạm phi ngữ cảnh**

* Văn phạm phi ngữ cảnh giới hạn các luật sinh phải có dạng A → α trong đó A ∈ Δ (nói một cách vắn tắt là vế trái của luật chỉ có 1 kí hiệu)

1. **Automat đẩy xuống**

* Automat đẩy xuống chuyên dùng để đoán nhận lớp ngôn ngữ phi ngữ cảnh
* Cấu trúc của automat gồm:

+ Bảng chuyển

+ Đầu đọc

+ Ngăn xếp

+Xâu vào

* Hoạt động của automat:

+ Bắt đầu từ trạng thái xuất phát

+ Đọc dữ liệu từ xâu vào

+ Quan sát bảng chuyển và ngăn xếp để biết sẽ xử lý thế nào

+ Dừng khi kết thúc xâu vào hoặc ở trạng thái kết thúc

1. **Văn phạm đệ quy trái**

* Văn phạm G gọi là văn phạm có đệ quy trái nếu chứa các luật dạng A → Aα | β
* Khử đệ quy trái

Ví dụ với luật trên: A → Aα | β

* Ta thêm kí hiệu trung gian mới R
* Và sửa luật thành: A → β R

R → α R | ε

1. **Văn phạm đơn nghĩa**

* Một văn phạm bị gọi là nhập nhằng (ambiguity) nếu tồn tại chuỗi w có ít nhất hai cây phân tích tạo ra nó
* Văn phạm không có nhập nhằng là văn phạm đơn nghĩa

1. **Suy dẫn**

* Khái niệm: αAβ ⇒ αγβ (gọi là αAβ suy dẫn ra αγβ) nếu A → γ là một luật sinh, α và β là các chuỗi ký hiệu thuộc ngôn ngữ L nào đó
* Nếu α 1 ⇒ α 2 ⇒ … ⇒ α n ta nói α 1 suy dẫn ra α n
* Hệ thống kí hiệu: ⇒ suy dẫn ra qua 1 bước (suy dẫn trực tiếp)

+ ⇒\* suy dẫn ra qua 0 hoặc nhiều bước

+ ⇒+ suy dẫn ra qua 1 hoặc nhiều bước

* Một số tính chất:

+ α ⇒\* α với ∀α

+ α ⇒\* β và β ⇒\* γ thì α ⇒\* γ

1. **Cây phân tích**

* Cây phân tích thể hiện cấu trúc của một suy dẫn

+ Nút gốc là kí hiệu bắt đầu

+ Các nút lá luôn là kí hiệu kết thúc

+ Các nút trong luôn là các kí hiệu trung gian

+ Cây không thể hiện thứ tự thực hiện các suy dẫn trực tiếp

• Việc duyệt cây sẽ tạo thành thứ tự thực hiện suy dẫn

• Suy dẫn trái tương đương với quá trình duyệt cây theo thứ tự giữa-trái-phải

1. **Top-down**

* Mục tiêu: trong số nhiều suy dẫn dạng S ⇒\* w, thuật toán sẽ tìm suy dẫn trái (hoặc suy dẫn phải)

1. **Bottom-up**

* Mục tiêu: trong số nhiều suy dẫn dạng S ⇒\* w, thuật toán sẽ tìm suy dẫn phải (hoặc suy dẫn trái)

1. **First(X)**

* Nếu X là kí hiệu kết thúc thì FIRST(X) là {X}
* Nếu X → ε là một luật sinh thì thêm ε vào FIRST(X)
* Nếu X → Y1Y2Y3 ...Yk là một luật sinh thì:

+ Thêm tất cả các ký hiệu kết thúc khác ε của FIRST (Y1) vào FIRST(X)

+ Nếu ε ∈ FIRST (Y1) thì tiếp tục thêm vào FIRST(X) tất cả các ký hiệu kết thúc khác ε của FIRST (Y2)

+ Nếu ε ∈ FIRST (Y1) ∩ FIRST (Y2) thì thêm tất cả các ký hiệu kết thúc khác ε ∈ FIRST (Y3)

+ Tiếp tục như vậy cho tới Yk

+ Thêm ε vào FIRST(X) nếu ε ∈ ∩i=1→k FIRST (Yi)

1. **Follow(A)**

* Tính FOLLOW (A): áp dụng các quy tắc sau cho đến khi không thể thêm gì vào mọi tập FOLLOW được nữa

+ Ðặt $ vào follow(S), trong đó S là ký hiệu bắt đầu của văn phạm và $ là ký hiệu kết thúc chuỗi nhập

+ Nếu có một luật sinh A→ αBβ thì thêm mọi phần tử khác ε của FIRST(β)vào trong FOLLOW(B)

+ Nếu có luật sinh A→ αB hoặc A→ αBβ mà ε ∈ FIRST(β) thì thêm tất cả các phần tử trong FOLLOW(A) vào FOLLOW(B)

1. **Bảng phân tích M**

1. Với mỗi luật sinh A→ α của văn phạm, thực hiện:

* Với mỗi ký hiệu kết thúc a ∈ FIRST(α), thêm A→ α vào M [A, a]
* Nếu ε ∈ FIRST(α) thì đưa luật sinh A→ α vào M [A, b] với mỗi ký hiệu kết thúc b ∈ FOLLOW(A)
* Nếu ε ∈ FIRST(α) và $ ∈ FOLLOW(A) thì đưa luật sinh A→ α vào M [A, $]
* Các ô trống trong bảng tương ứng với lỗi (error)