BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC ĐÀ NẴNG

TRẦN THỊ BÍCH HẰNG

NGHIÊN CỨU ỨNG DỤNG ANTLR ĐỂ XỬ LÝ CÁC THÔNG ĐIỆP TRONG PHẦN MỀM ĐA NGỮ

Chuyên ngành: Khoa học máy tính
Mã số: 60.48.01

TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT

Đà Nẵng - Năm 2013

Công trình được hoàn thành tại

ĐẠI HỌC ĐÀ NẴNG

Người hướng dẫn khoa học: **PGS.TS. VÕ TRUNG HÙNG**

Phản biện 1 : **TS. HUỲNH HỮU HƯNG**

Phản biện 2 : **PGS.TS. ĐOÀN VĂN BAN**

Luận văn được bảo vệ trước Hội đồng chấm Luận văn tốt nghiệp thạc sĩ Kỹ thuật họp tại Đại học Đà Nẵng vào ngày 19 tháng 5 năm 2013.

Có thể tìm hiểu luận văn tại:

- Trung tâm Thông tin - Học liệu, Đại Học Đà Nẵng

MỞ ĐẦU

1. Lý do chọn đề tài

Trong nhận dạng ngôn ngữ dựa trên máy tính, ANTLR (viết tắt cho ANother Tool for Language Recognition) là một bộ phân tích cú pháp dựa trên phân tích LL(k). Phiên bản trước đó của ANTLR là bộ phân tích cú pháp PCCTS (viết tắt của Purdue Compiler Construction Tool Set), được phát triển lần đầu vào năm 1989 và hiện vẫn đang được phát triển.

Trong đề tài này, tôi nghiên cứu việc ứng dụng ALTLR để thử nhiệm các công cụ của ANTLR để xử lý các thông điệp đa ngữ. Đây là 2 lĩnh vực khá mới và chưa được nghiên cứu, phổ biến rộng rãi ở Việt Nam. Vì vậy, tôi chọn đề tài *"Nghiên cứu ứng dụng ANTLR để xử lý các thông điệp trong phần mềm đa ngữ"* làm đề tài tốt nghiệp cao học của mình.

2. Mục tiêu và nhiệm vụ nghiên cứu

Mục tiêu của đề tài là ứng dụng ngôn ngữ đặc tả và phân tích cú pháp ANTLR vào xử lý các thông điệp đa ngữ.

Nhiệm vụ chính của đề tài bao gồm: nghiên cứu cơ sở lý thuyết về đặc tả hình thức, nghiên cứu cơ sở lý thuyết về chương trình dịch, nghiên cứu ngôn ngữ ANTLR, nghiên cứu cơ sở lý thuyết và thư viện GetAMsg và sau đó vận dụng ANTLR vào quản lý các thông điệp đa ngữ sử dụng trong GetAMsg.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu bao gồm: các vấn đề liên quan đến đặc tả hình thức, các vấn đề liên quan đến chương trình dịch, ngôn ngữ ANTLR và ứng dụng của nó.

Phạm vi nghiên cứu là dùng ngôn ngữ ANTLR để xử lý các thông điệp của GetAMsg.

4. Phương pháp nghiên cứu

Trong quá trình thực hiện, chúng tôi sử dụng hai phương pháp chính là nghiên cứu tài liệu và thực nghiệm. Với phương pháp đầu tiên, chúng tôi tiến hành thu thập và nghiên cứu các tài liệu có liên quan đến đề tài. Phương pháp tiếp theo là nghiên cứu các công cụ ANTLR sẵn có, tiến hành thử nghiệm. Cuối cùng là đánh giá kết quả và nêu hướng phát triển của đề tài.

5. Ý nghĩa khoa học và thực tiễn của đề tài

Đề tài mang ý nghĩa khoa học góp phần nghiên cứu và phổ biến hai công cụ khá mới mẻ ở Việt Nam là ANTLR và GetAMsg. Đây sẽ là tài liệu tham khảo quan trọng cho sinh viên, học viên cao học, nghiên cứu sinh về xử lý ngôn ngữ tự nhiên và đa ngữ hóa các phần mềm.

Ngoài ra đề tài mang ý nghĩa thực tiễn: ANTLR là công cụ ngôn ngữ cung cấp và tự xây dựng trình biên dịch, tự động hóa trong việc kiểm tra tính đúng đắn về ngữ pháp, từ vựng. Đây là kỹ thuật chưa được nghiên cứu và phổ biến ở Việt Nam, điều đó mở ra hướng nghiên cứu, ứng dụng mới.

6. Cấu trúc luận văn

Cấu trúc luận văn được tổ chức thành 3 chương. Chương đầu giới thiệu phần nghiên cứu tổng quan về đặc tả phần mềm và chương trình dịch. Chương hai là giới thiệu ngôn ngữ ANTLR và các ứng dụng của nó. Chương ba thử nhiệm các công cụ của ANTLR để xử lý các thông điệp đa ngữ. Cuối cùng là kết luận của đề tài.

CHƯƠNG 1
NGHIÊN CỨU TỔNG QUAN VỀ ĐẶC TẢ PHẦN MỀM, TRÌNH BIÊN DỊCH, PHẦN MỀM ĐA NGỮ

1.1 TỔNG QUAN VỀ ĐẶC TẢ PHẦN MỀM

Việc cạnh tranh của ngành du lịch có thể được minh họa trên hai phương diện, trong nước và quốc tế.

1.1.1 Khái niệm đặc tả phần mềm

Đặc tả (specification) được định nghĩa trong từ điển tiếng Việt (1997): *Mô tả thật chi tiết một bộ phận đặc biệt tiêu biểu để làm nổi bật bản chất của toàn thể".*

Theo Computer Dictionary của Microsoft Press (1994), đặc tả là *sự mô tả chi tiết: Về mặt phần cứng, đặc tả cung cấp thông tin về các thành phần, khả năng và yếu tố kỹ thuật của máy tính. Về mặt phần mềm, đặc tả mô tả môi trường hoạt động và chức năng của chương trình*.

1.1.2 Các phương pháp đặc tả

Có thể chia đặc tả yêu cầu ra làm hai loại: đặc tả phi hình thức (ngôn ngữ tự nhiên) và đặc tả hình thức (dựa trên kiến trúc toán học).

1.2 TỔNG QUAN VỀ TRÌNH BIÊN DỊCH

1.2.1 Khái niệm trình biên dịch

Trình biên dịch là một chương trình làm nhiệm vụ đọc một chương trình được viết bằng một ngôn ngữ - ngôn ngữ nguồn (source language) rồi dịch nó thành một chương trình tương đương ở một ngôn ngữ khác - ngôn ngữ đích (target languague). Cấu trúc của trình bao gồm các giai đoạn: phân tích từ vựng; phân tích cú pháp; phân tích ngữ nghĩa; sinh mã trung gian; tối ưu mã và sinh mã đích.

1.2.2 Mô hình phân tích- tổng hợp của một trình biên dịch

Chương trình dịch thường bao gồm 2 quá trình: phân tích và tổng hợp

- Phân tích → đặc tả trung gian

- Tổng hợp → chương trình đích

Trong quá trình phân tích chương trình nguồn sẽ được phân rã thành một cấu trúc phân cấp, thường là dạng cây - cây cú pháp (syntax tree) mà trong đó có mỗi nút là một toán tử và các nhánh con là các toán hạng.

1.2.3 Môi trường của trình biên dịch

Một chương trình nguồn có thể được phân thành các module và được lưu trong các tập tin riêng rẻ. Công việc tập hợp lại các tập tin này thường được giao cho một chương trình riêng biệt gọi là bộ tiền xử lý (preprocessor). Bộ tiền xử lý có thể "bung" các ký hiệu tắt được gọi là các macro thành các câu lệnh của ngôn ngữ nguồn.

1.2.4 Các giai đoạn biên dịch

Một trình biên dịch được chia thành các giai đoạn: phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa, sinh mã trung gian, tối ưu mã, sinh mã đích.

1.2.5 Phân tích từ vựng (Lexical Analysis)

Phân tích từ vựng là giai đoạn đầu tiên của mọi trình biên dịch. Nhiệm vụ chủ yếu của nó là đọc các ký hiệu đầu vào rồi tạo ra một chuỗi các mã thông báo token được sử dụng bởi bộ phân tích cú pháp. Sự tương tác này được thể hiện như hình sau, trong đó bộ phân tích từ vựng được thiết kế như một thủ tục được gọi bởi bộ phân tích cú pháp, trả về một mã thông báo khi được gọi.

1.2.6 Phân tích cú pháp (Syntax Analysis)

Giai đoạn phân tích cú pháp thực hiện công việc nhóm các thẻ từ của chương trình nguồn thành các ngữ đoạn văn phạm (grammatical phrase), mà sau đó sẽ được trình biên dịch tổng hợp ra thành phẩm. Thông thường, các ngữ đoạn văn phạm này được biểu diễn bằng dạng cây phân tích cú pháp (parse tree) với:

- Ngôn ngữ được đặc tả bởi các luật sinh.
- Phân tích cú pháp dựa vào luật sinh để xây dựng cây phân

tích cú pháp.

1.2.7 Phân tích ngữ nghĩa (Semantic Analysis)

Giai đoạn phân tích ngữ nghĩa sẽ thực hiện việc kiểm tra xem chương trình nguồn có chứa lỗi về ngữ nghĩa hay không và tập hợp thông tin về kiểu cho giai đoạn sinh mã về sau. Một phần quan trọng trong giai đoạn phân tích ngữ nghĩa là kiểm tra kiểu (type checking) và ép chuyển đổi kiểu.

1.2.8 Sinh mã trung gian

Sau khi phân tích ngữ nghĩa, một số trình biên dịch sẽ tạo ra một dạng biểu diễn trung gian của chương trình nguồn. Chúng ta có thể xem dạng biểu diễn này như một chương trình dành cho một máy trừu tượng. Chúng có 2 đặc tính quan trọng: dễ sinh và dễ dịch thành chương trình đích.

1.2.9 Tối ưu mã

Giai đoạn tối ưu mã cố gắng cải thiện mã trung gian để có thể có mã máy thực hiện nhanh hơn. Một số phương pháp tối ưu hóa hoàn toàn bình thường.

Có một khác biệt rất lớn giữa khối lượng tối ưu hoá mã được các trình biên dịch khác nhau thực hiện. Trong những trình biên dịch gọi là "trình biên dịch chuyên tối ưu", một phần thời gian đáng kể được dành cho giai đoạn này.

1.2.10 Sinh mã đích

Giai đoạn cuối cùng của biên dịch là sinh mã đích, thường là mã máy hoặc mã hợp ngữ. Các vị trí vùng nhớ được chọn lựa cho mỗi biến được chương trình sử dụng. Sau đó, các chỉ thị trung gian được dịch lần lượt thành chuỗi các chỉ thị mã máy. Vấn đề quyết định là việc gán các biến cho các thanh ghi.

1.3 TỔNG QUAN VỀ PHẦN MỀM ĐA NGỮ

1.3.1 Bối cảnh về toàn cầu hóa các phần mềm

Trước đây, các phần mềm, trang Web thường được viết bằng

tiếng Anh nhưng trên thực tế có rất nhiều người trên thế giới không thể đọc và hiểu được tiếng Anh. Có nhiều người tuy sử dụng được tiếng Anh nhưng họ vẫn yêu thích làm việc trên những phần mềm và website trong ngôn ngữ mẹ đẻ (native language) và muốn chuyển sang ngôn ngữ tiếng Anh mất nhiều thời gian và công sức. Do đó nhu cầu về đa ngữ hóa phần mềm ra đời.

1.3.2 Khái niệm phần mềm đa ngữ

Phần mềm đa ngữ (Multilingual Software) là phần mềm cho phép người sử dụng chọn lựa ngôn ngữ (ví dụ: tiếng Anh, tiếng Pháp, tiếng Việt...) khi làm việc với phần mềm đó. Người sử dụng sẽ chọn lựa ngôn ngữ sử dụng cho mình, hiện nay, đã có những phần mềm đáp ứng được nhu cầu này nhưng chưa thực sự nhiều.

1.3.3 Nhu cầu về đa ngữ hóa phần mềm

Đa dạng hóa ngôn ngữ trong các sản phẩm phần mềm là xu hướng của các nhà sản xuất phần mềm hiện nay, bởi tính toàn cầu hóa cũng như phương hướng mở rộng thị trường ra quốc tế. Lựa chọn đối tác dịch chuyên nghiệp là phương pháp mà các nhà sản xuất phần mềm sử dụng.

Nhu cầu đa ngữ hóa phần mềm đặt ra một cách tự nhiên. Cùng với quá trình toàn cầu hóa, việc đa ngữ hóa và bản địa hóa phần mềm đang là vấn đề được các nhà khoa học, các nhà sản xuất phần mềm quan tâm.

1.3.4 Các mô hình tổ chức quản lý các thông điệp

Có 3 mô hình tổ chức phần mềm đa ngữ được sử dụng hiện nay là:

Mô hình 1: Phương pháp truyền thống là các thông điệp được viết gắn liền với mã nguồn chương trình. Với mô hình nay, người ta viết chương trình cho tiếng Anh, sau đó tạo ra một bản sao và sửa lại cho tiếng Pháp...

Mô hình 2: Phương pháp tách rời mã nguồn chương trình với các dữ liệu ngôn ngữ. Các dữ liệu ngôn ngữ (thông điệp, các hàm...)

đặt trong một tập tin dùng cho tất cả các ngôn ngữ.

Mô hình 3: Sử dụng phương pháp tách rời mã nguồn với các dữ liệu ngôn ngữ và dữ liệu của các ngôn ngữ là lưu trữ riêng biệt cho mỗi ngôn ngữ, ứng với một ngôn ngữ là một tập tin chứa dữ liệu của ngôn ngữ đó.

1.3.5 Một số công cụ quản lý thông điệp đa ngữ

Hiện tại có những hệ thống như Catgets, Ariane-G5, Gettext, GetAMsg cho phép "bản địa hóa" (localisation) dễ dàng những tập tin thông điệp nhưng chứng chưa thế xử lý những biến đổi ngôn ngữ bên trong những thông điệp.

1.4 BỘ CÔNG CỤ GETTEXT

1.4.1 Giới thiệu Gettext

Gettext: là một phần mềm mã nguồn mở phục vụ bản địa hóa phần mềm và được dùng cho một số ngôn ngữ thông dụng. Gettext hỗ trợ dịch những thông điệp từ một ngôn ngữ nào đó ra ngôn ngữ mà người sử dụng có thể đọc được ngay. Hiện tại Gettext hỗ trợ bản địa hóa cho 12 ngôn ngữ khác nhau và mỗi ngôn ngữ sử dụng một hệ thống mã hóa khác nhau hay có thể chuyển mã về một hệ thống mã duy nhất là UTF-8 (Unicode Transformation Format 8- Bit).

1.4.2 Quy trình xử lý trong Gettext

Trong quá trình đa ngữ hóa, ta chủ yếu làm việc với tập tin:

- Tập tin POT (Portable Object Template): bước đầu tiên file này được tạo ra khi chúng ta dùng phần mềm hay script để quét mã nguồn dựa vào các marked function bên trên. Mục đích chính là lấy ra các language mặc định trong mã nguồn.

- Tập tin PO (Portable Object): đây là bước ta sẽ dịch toàn bộ language đã quét được ở bước 1 sang ngôn ngữ mà chúng ta muốn.

- MO (Machine Object) files: bước cuối cùng là dịch file PO sang mã máy để tối ưu cho việc sử dụng.

1.4.3 Tạo tập tin mẫu PO

Sau khi chuẩn bị xong nguồn, tiến hành tạo tập tin mẫu PO, phần này sẽ giải thích cách sử dụng xgettext cho mục đích này. Xgettext sẽ tạo ra một tập tin có tên domainname.po, ta nên đổi tên nó thành domainname.pot.

1.4.4 Tạo tập tin .PO mới

Khi bắt đầu một bản dịch mới, sẽ tạo một tập tin là LANG.po,như là một bản sao của tập tin mẫu package.pot. Cách dễ nhất để thực hiện việc này đó là sử dụng chương trình 'msginit'. Chương trình msginit tạo ra một tập tin .PO mới, khởi tạo các thông tin với giá trị được lấy từ môi trường của người sử dụng. Nếu không có tập tin inputfile nào, thư mục hiện tại sẽ tìm kiếm tập tin .POT.

1.4.5 Cập nhật tập tin PO, gọi chương trình msgmerge

Thêm thư mục directory vào danh sách các thư mục. Tập tin nguồn sẽ được tìm kiếm trong danh sách các thư mục này, kết quả là tập tin .po sẽ được ghi vào thư mục hiện hành.

1.4.6 Tạo tập tin .MO

Offset O và offset T là hai bảng mô tả chuỗi, mỗi mô tả chuỗi sử dụng 32 bít số nguyên, để đếm các byte đầu của tập tin. Bảng đầu tiên chứa các mô tả về chuỗi chuẩn, và sắp xếp các chuỗi này tăng dần theo thứ tự trong từ điển. Bảng thứ hai chứa các mô tả về các chuỗi đã được dịch và nó song song với bảng thứ nhất để tìm các chuỗi dịch tương ứng cho phép truy cập tới mảng trên mảng thứ hai với cùng chỉ số.

1.5 BỘ CÔNG CỤ GETAMSG

1.5.1 Mô hình hóa những thông điệp có chứa biến nhớ và biến thể

Trong những chương trình máy tính, chúng ta thường gặp những thông điệp (message) có chứa những biến nhớ (variable). Những biến nhớ này kéo theo những biến đổi bên trong thông điệp, nguyên nhân tạo ra những biến đổi này là do sự tương hợp về giống,

số, cách, mức độ lịch sự và vị trí của các biến…

1.5.2 Biểu diễn các thông điệp trong chương trình

Dành cho tiếng Anh và tiếng Pháp, biến &c là được định nghĩa bởi điều kiện $n<=1. Ở đó, hai biến thể (dưới định dạng có thể sử dụng) tương ứng với 2 khả năng :

Bảng 1.1 : *Những biến thể của thông điệp trong tiếng Anh và tiếng Pháp*

&c	Điều kiện	Tiếng Anh	Tiếng Pháp
1	$n <= 1	$n file has been removed	$n fichier a été supprimé
2	Sinon	$n files have been removed	$n fichiers ont été supprimés

1.5.3 Phát triển một công cụ quản lý thông điệp

Chúng ta thực hiện một sự phân ly hoàn toàn giữa mã chương trình (code source) và những thông điệp của chương trình này (qua công cụ GetAMsg). Để thực hiện điều này, người ta có thể sử dụng những công cụ của gettext để trích những thông điệp ra khỏi chương trình nguồn. Một tập tin thông điệp (ví dụ : FicMes1_fre.adm) chứa những thông điệp, hoặc chính xác hơn hơn là những ô-tô-mát của các thông điệp, dành cho một ngôn ngữ cho trước (ở đây là tiếng Pháp).

1.5.4 Dịch các tập thông điệp

Chúng ta quản lý trước hết (trong ngôn ngữ hiện có – ngôn ngữ nguồn) một danh sách những giá trị có thể tương ứng với các "trường hợp" có thể trong ngôn ngữ cần dịch sang (ngôn ngữ đích), trong khi vẫn bảo toàn các biến (ví dụ: $n_night=2). Tiếp theo, chúng ta dịch những "trường hợp" này sang ngôn ngữ đích. Cuối cùng, chúng ta thiết lập lại một ô-tô-mát thông điệp trong ngôn ngữ đích từ tập hợp những "trường hợp" có được trong bước biểu diễn các thông điệp chương trình.

CHƯƠNG 2
CÔNG CỤ ANTLR

2.1 KHÁI QUÁT ANTLR

2.1.1 Khái niệm ANTLR

ANTLR (ANother Tool for Language Recognition) là một công cụ nhận dạng ngôn ngữ cung cấp, xây dựng trình biên dịch từ các mô tả ngữ pháp có chứa các hành động trong một loạt các ngôn ngữ mục tiêu. ANTLR tự động hóa xây dựng nhận dạng ngôn ngữ. Từ một ngữ pháp chính quy, ANTLR tạo ra một chương trình xác định xem câu phù hợp với ngôn ngữ đó. Nói cách khác, đó là một chương trình để viết các chương trình khác bằng cách thêm vào các đoạn mã để ngữ pháp, nhận diện trở thành một phiên dịch hay thông dịch viên. ANTLR cung cấp hỗ trợ tuyệt vời cho cây phân tích cú pháp trừu tượng (AST-Abstract Syntax Tree), hình thức xây dựng dịch thuật, và sử dụng các phần mềm dịch tự động.

2.1.2 Chức năng của ANTLR

ANTLR đọc một tập tin ngôn ngữ mô tả được gọi là một ngữ pháp và tạo ra một số các tập tin mã nguồn và các tập tin phụ trợ khác. ANTLR tạo ra 2 công cụ:

- **Phân tích từ vựng (Lexer):** đọc một ký tự đầu vào hoặc byte dòng (tức là ký tự, dữ liệu nhị phân, ...), phân chia token bằng cách sử dụng các mẫu chỉ định và tạo ra một dòng token như đầu ra. Nó cũng có thể đánh dấu một số token như khoảng trắng và là ẩn bằng cách sử dụng một giao thức phân tích cú pháp ANTLR.

- **Phân tích cú pháp (Parser):** là quá trình phân tích một văn bản, được thực hiện của một chuỗi các token, để xác định cấu trúc ngữ pháp của nó đối với một (nhiều hơn hoặc ít hơn) ngữ pháp chính quy. Phân tích cú pháp cũng có thể được sử dụng như một thuật ngữ ngôn ngữ học.

2.1.3 Ưu và nhược điểm của ANTLR

a. Ưu điểm

ANTLR mang lại những tính năng ưu việt như:

- Tự động tạo trình phân tích từ vựng và trình phân tích cú pháp;
- Tự động sinh mã;
- Linh hoạt và xử lý lỗi hiệu quả;
- ANTLR đi kèm với mã nguồn mở hoàn toàn không giống như nhiều hệ thống khác và hoàn toàn không hạn chế về việc sử dụng nó.

b. Nhược điểm

ANTLR là một bộ phân tích cú pháp dựa trên phân tích LL(k) còn Yacc và Bison là hai bộ phân tích cú pháp tương tự lại dựa trên phân tích cú pháp LALR (Look-Ahead Left Right). Trong đó, phân tích cú pháp LALR là công cụ đọc BNF ngữ pháp và tạo ra phần mềm phân tích ngôn ngữ LALR có khả năng phân tích cú pháp tập tin viết ngôn ngữ máy tính định rõ bằng ngữ pháp BNF (Backus–Naur Form) để biểu diễn văn phạm phi ngữ cảnh. LALR sẽ phân tích nhanh hơn vì nó xử lý theo chế độ bảng, ANTLR sử dụng dòng dõi phân tích cú pháp đệ quy do đó hơi chậm so với Yacc và Bison.

2.1.4 Phiên bản mới nhất của ANTLR

ANTLR 3 là phiên bản mới nhất của một bộ công cụ xử lý ngôn ngữ đầu tiên được phát hành như là PCCTS trong giữa những năm 1990. Ngữ pháp của ANTLR mô tả tùy chọn có thể bao gồm các mã code văn bản hay còn được gọi là ngôn ngữ mục tiêu. Các ngôn ngữ mục tiêu bao gồm Java, C #, Objective C, C, Python và Ruby ngoài ra còn có C++, Perl6 và Oberon.

ANTLR 3 có thể giúp chúng ta tiết kiệm thời gian và nguồn lực bằng cách tự động xây dựng các công cụ xử lý ngôn ngữ. Nó cũng là các công cụ sinh ra các trình biên dịch của trình biên dịch có một

tác động tích cực về hiệu quả phát triển. Ngoài ra, nhiều tính năng mới của ANTLR 3 bao gồm một công cụ phân tích được cải tiến, sức mạnh tăng cường đáng kể khả năng phân tích cú pháp của nó thông qua phân tích cú pháp LL.

2.1.5 Các phần mềm cần thiết

- Phần mềm Java JDK - Java Development Kit (không phải là JRE).

- Phần mềm ANTLR 4.0 giải nén chuyển vào thư mục lib lưu ổ đĩa C.

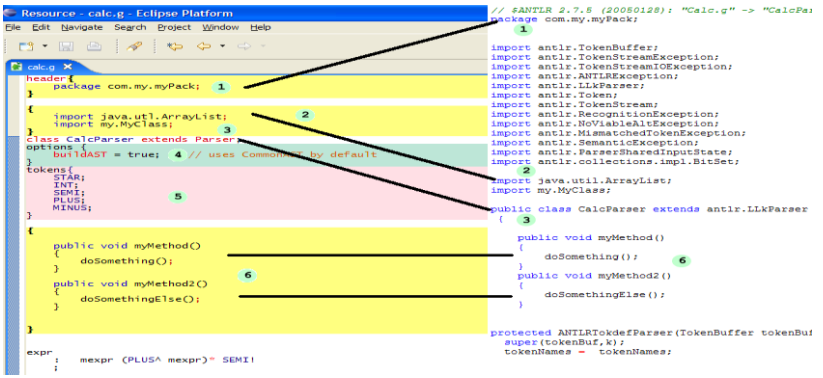
- Eclipse Galileo giải nén chuyển vào thư mục lib lưu ổ đĩa C.

- Các công cụ Dynamic Languages Toolkit Core Frameworks (DLTK), Graphical Editing Framework (GEF) và Graphical Editing Framework Zest Visualization Toolkit SDK (Zest) được cài đặt vào Eclipse.

2.1.6 Tập tin ngữ pháp và quy tắc của ANTLR

a. Tập tin ngữ pháp của ANTLR

Chúng ta có nhầm lẫn bởi tất cả các phần khác nhau của một tập tin ngữ pháp ANTLR, hình sau chỉ dẫn rõ các tập tin ngữ pháp của ANTLR:



Hình 2.1 : *Các ngữ pháp của ANTLR*

(1) Tất cả những gì chúng ta đặt trong phần tiêu đề của các tập tin ngữ pháp được đặt ngay trên đầu trang của tất cả các file java tạo ra bởi ANTLR. Chúng ta thường sử dụng phần này để đặt định nghĩa gói.

(2) Phần câu lệnh dưới đó là duy nhất cho mỗi ngữ pháp chúng ta chỉ định trong tập tin.

(3) Sau đó, chúng tôi có các đặc điểm kỹ thuật ngữ pháp, thậm chí trông giống như một khai báo lớp.

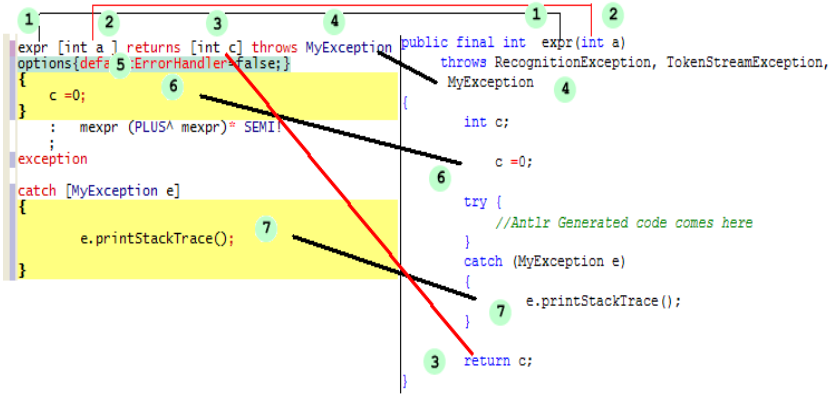
(4) Dưới đây là phần tùy chọn, nơi chúng ta có thể chỉ định các tùy chọn cho ngữ pháp. Chúng ta có thể nhấn Ctrl + Space trong ANTLR Studio để xem các tùy chọn có sẵn.

(5) Phần token được sử dụng để xác định các token "ảo" mà không được khai báo trong lexer. Đây thường được sử dụng trong TreeParsers để xác định các token "ảo".

(6) Đặt công cụ của nó "bên trong" các định nghĩa lớp. Chúng ta thường sử dụng nó để xác định một số phương pháp tùy chỉnh cho phân tích cú pháp.

b. Quy tắc của ANTLR

Xem quy định theo hình vẽ sau:



Hình 2.2 : *Các quy tắc của ANTLR*

(7) (2) (3) (4) Các lệnh được thực hiện với một hàm. Chúng ta có thể chỉ định đối số cho quy tắc, như được hiển thị ở trên (int a), ngay cả một giá trị trả về và các trường hợp ngoại lệ không tuân thủ quy tắc.

(8) Phần tùy chọn cho phép chúng ta xác định một số tùy chọn cụ thể quy tắc.

(9) Lời gọi giá trị.

(10) Chúng ta có thể xử lý tùy chỉnh ngoại lệ trong phần ngoại lệ.

2.1.7 ANTLRWorks

ANTLRWorks cho phép người dùng chỉnh sửa, hình dung, giải thích và gỡ lỗi bất kỳ thông qua một giao diện dễ sử dụng, đồ họa người dùng. Nó kết hợp một trình soạn thảo ngữ pháp nhận thức tuyệt vời với một thông dịch viên cho tạo mẫu nhanh và một trình gỡ lỗi ngôn ngữ để cô lập lỗi ngữ pháp.

2.2 CÁC TÍNH NĂNG CỦA ANTLR

2.2.1 Định nghĩa lớp phân tích cú pháp (Parser Class Definitions)

Một lớp class tùy chọn là một số text tùy ý kèm theo trong {}. Kèm theo dấu ngoặc nhọn {} không được sử dụng để phân định các lớp class bởi vì nó là khó để kết hợp dấu ngoặc phải ở dưới cùng của một tập tin với ngoặc bên trái ở phía trên cùng của tập tin. Thay vào đó, một lớp phân tích cú pháp được giả định tiếp tục cho đến khi xuất hiện lớp tiếp theo.

2.2.2 Định nghĩa lớp phân tích từ vựng (Lexical Analyzer Class Definitions)

Một lớp phân tích cú pháp trong các đối tượng phân tích cú pháp biết làm thế nào để áp dụng các cấu trúc ngữ pháp liên quan đến một dòng đầu vào của thẻ token. Để thực hiện những phân tích từ vựng cần phải chỉ định một lớp lexer mô tả làm thế nào để phá vỡ các dòng ký tự nhập vào thành một dòng của thẻ token. Cú pháp là

tương tự như của một lớp phân tích cú pháp.

2.3 PHÂN TÍCH TỪ VỰNG VỚI ANTLR

2.3.1 Quy tắc từ vựng Lexcial

Quy tắc được định nghĩa trong một ngữ pháp lexer phải có tên bắt đầu với một ký tự hoa. Quy tắc lexer được xử lý trong cùng một cách thức chính xác như các quy tắc phân tích cú pháp và do đó, có thể chỉ định các đối số và giá trị trả lại, các lexer quy tắc cũng có thể có các biến địa phương (local variables) và sử dụng đệ quy. Các quy tắc sau đây xác định một quy tắc được gọi là ID đó là có sẵn như là một loại mã thông báo trong phân tích cú pháp.

2.3.2 Bỏ qua ký tự

Sử dụng quy tắc bỏ qua rule ignored để có một ký tự phù hợp, thiết lập các loại Token.SKIP. Bỏ qua token có hiệu lực cho lexer để thiết lập lại và cố gắng cho token khác. Token bỏ qua không bao giờ được gửi trở lại để phân tích cú pháp.

2.3.3 Trả lại giá trị

Tất cả các quy tắc tự động trả về một đối tượng mã thông báo (theo lý thuyết) trong đó có các nội dung văn bản text phù hợp cho các quy tắc và các loại token. Để chỉ định user-defined xác định giá trị trả lại, xác định một giá trị trả về.

2.3.4 Từ khóa (Keywords) và chuỗi ký tự (Literals)

Nhiều ngôn ngữ có chung nguyên tắc là "nhận diện" từ vựng và từ khóa là trường hợp đặc biệt của các mô hình nhận dạng. Điều này thường mâu thuẫn với từ khoá. ANTLR giải quyết vấn đề này bằng cách cho phép đặt từ khóa cố định vào một bảng chữ. Bảng chữ (luôn được thực thi như một bảng băm hash table trong lexer) được kiểm tra sau khi mỗi token phù hợp, vì vậy để các literals hiệu quả ghi đè lên các mô hình nhận dạng tổng quát hơn.

2.3.5 Quét file nhị phân

Chuỗi ký tự không giới hạn ký tự ASCII. Khi phân tích một tập

tin nhị phân có chứa chuỗi và số nguyên ngắn. Đầu tiên, xác định các lớp và thiết lập từ vựng để có tất cả 8 bit nhị phân giá trị.

2.3.6 Quét các ký tự Unicode

ANTLR cho phép nhận ra đầu vào bao gồm các ký tự Unicode, có nghĩa là không bị giới hạn đến 8 bit ký tự ASCII. ANTLR cho phép nhưng không hỗ trợ Unicode.

2.3.7 Tạo đối tượng Token

Các quy tắc từ vựng có thể gọi các quy tắc khác giống như trong phân tích cú pháp. ANTLR cho phép dán nhãn các quy tắc từ vựng và có được một đối tượng token đại diện cho văn bản, loại thẻ, số dòng, vv .. phù hợp với quy tắc tham chiếu.

2.3.8 Điều kiện End of file

Một phương pháp có sẵn để tác động trở lại với kết thúc của tình trạng tập tin như thể nó là một sự kiện. Phương pháp này, CharScanner.uponEOF (), được gọi là từ nextToken () ngay trước khi máy quét trả về một đối tượng EOF_TYPE mã thông báo để phân tích cú pháp.

2.4 CÂY PHÂN TÍCH CÚ PHÁP ANTLR

2.4.1 Quy tắc ngữ pháp của cây

Ngữ pháp cây là bộ sưu tập các quy tắc EBNF nhúng với hành động, vị ngữ nghĩa, và các vị cú pháp. Mỗi phương án thay thế bao gồm một danh sách các yếu tố, một yếu tố có thể là một trong các mục trong một ngữ pháp ANTLR thường xuyên với việc bổ sung các yếu tố mô hình cây.

2.4.2 Cú pháp vị từ

ANTLR cây phân tích cú pháp sử dụng một biểu tượng duy nhất của lookahead, là hình thức trung gian được thiết kế một cách rõ ràng dễ dàng. Tuy nhiên cần phải phân biệt giữa các cấu trúc cây tương tự. Vị cú pháp có thể được sử dụng để khắc phục những hạn chế của giới hạn cố định lookahead.

2.4.3 Dòng Token

Theo truyền thống, một lexer và phân tích cú pháp chặt chẽ cùng các đối tượng. Tuy nhiên, công nhận ngôn ngữ và bản dịch có thể được hưởng lợi rất nhiều từ xử lý kết nối giữa lexer và phân tích cú pháp như là một dòng token. Tương tự dòng Java I/O rất nhiều đường truyền của các đối tượng dòng để sản xuất dòng dữ liệu bậc cao.

2.4.4 Lọc dòng Token

ANTLR cung cấp TokenStreamBasicFilter cho các tình huống như, vì vậy có thể hướng dẫn để loại bỏ bất kỳ loại thẻ hoặc các loại mà không cần phải sửa đổi các lexer. Đây là một cách sử dụng ví dụ của TokenStreamBasicFilter lọc ra dòng chú thích và khoảng trắng.

2.4.5 Sự phân tách dòng Token

Để gửi các comment phân tích cú pháp trên một dòng token thông báo ẩn - một trình phân tích cú pháp "lắng nghe" (listening). Trong quá trình công nhận, các hành động sau đó có thể kiểm tra các dòng ẩn, thu thập các ý kiến.

2.4.6 Cơ chế StringTemple

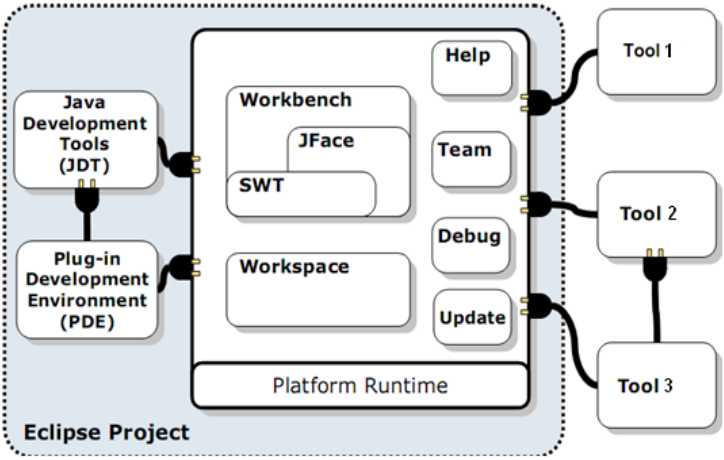
Cách tiếp cận StringTemplate là để tạo ra một cấu trúc dữ liệu trung gian (StringTemplate) trong cây phân tích cú pháp trừ tượng AST (Abstract Syntax Tree) và sau đó có khả năng có nhiều đầu ra cho các ngôn ngữ khác nhau, có những định dạng đầu ra rõ ràng tách ra từ phân tích và xây dựng các cấu trúc dữ liệu.

2.5 ECLIPSE

2.5.1 Giao diện chính của Eclipse

Cửa sổ chính của Eclipse được gọi là bàn làm việc (workbench), nó có thành phần như: thanh trình đơn, thanh công cụ, trình soạn thảo và các khung nhìn. Vùng phía dưới thanh công cụ nơi đặt trình soạn thảo và các khung nhìn được gọi là trang bàn làm việc (workbench page).

2.5.2 Kiến trúc của Eclipse



Hình 2.3 : *Kiến trúc của Eclipse*

Kiến trúc của Eclipse được xây dựng dựa trên hai thành phần:

- Thành phần nền tảng (core) : Có thể hình dung phần này bao gồm các chức năng, dịch vụ mà các hệ phát triển ứng dụng phải có như chức năng cung cấp giao diện, trình soạn thảo văn bản, hướng dẫn sử dụng, gỡ lỗi… cần cho mọi nền tảng lập trình (cần cho các plug-in).

- Thành phần gắn thêm (plug-in) : Nền tảng của Eclipse được xây dựng dựa trên cơ chế phát hiện, tích hợp và chạy đoạn chương trình (chính là thành phần gắn thêm). Nhờ cơ chế này mà nó gọn nhẹ và dễ tùy biến. Việc "gắn" thêm thành phần được thực hiện rất đơn giản, bằng việc chép các tập tin vào đúng thư mục, khi khởi động lại Eclipse tự động nhận được các thành phần gắn thêm.

2.5.3 Ưu điểm và nhược điểm của Eclipse

a. Ưu điểm

- Là mã nguồn mở miễn phí do đó nó có khả năng mở rộng, phụ thuộc vào các thành phần gắn thêm như cho ngôn ngữ mới, cho bộ xử lý mới;

- Ứng dụng được cho việc phát triển mọi kiểu ứng dụng, từ ứng dụng trong doanh nghiệp, ứng dụng trên máy tính cá nhân cho đến các ứng dụng nhúng cho các thiết bị;

- Lập trình viên có thể tự làm thêm các thành phần gắn thêm theo yêu cầu riêng của mình.

b. Nhược điểm

Người dùng cần có hiểu biết rõ về Eclipse để biết lúc nào thì gắn thêm hay gỡ ra dự án gì. Ngoài ra, các dự án (project) gắn thêm không ngừng phát triển nên phải biết lúc nào thì nâng cấp lên đời mới hơn (mặc dù Eclipse cũng có khả năng tự động tìm dự án gắn thêm).

CHƯƠNG 3
THỬ NGHIỆM CÁC CÔNG CỤ CỦA ANTLR ĐỂ XỬ LÝ CÁC THÔNG ĐIỆP ĐA NGỮ

3.1 CÀI ĐẶT ANTLR

Bước 1: Tải Java JDK (Java SE Development Kit)

Bước 2: Tải ANTLR 3.4 từ trang Web http://www.antlr.org/.

Bước 3: Tải Eclipse Galileo bằng cách vào trang Web http://eclipse.org/downloads, chọn Eclipse IDE for Java Developers.

Bước 4: Cài đặt DLTK, GEF và Zest bằng cách thêm vào đường link http://download.eclipse.org/releases/galileo ở mục Help/Install New Software .

Bước 5: Tải ANTLR IDE bằng cách thêm trang Web http://antlrv3ide.sourceforge.net/updates/2.0.2 vào.

3.2 THỬ NGHIỆM ANTLR

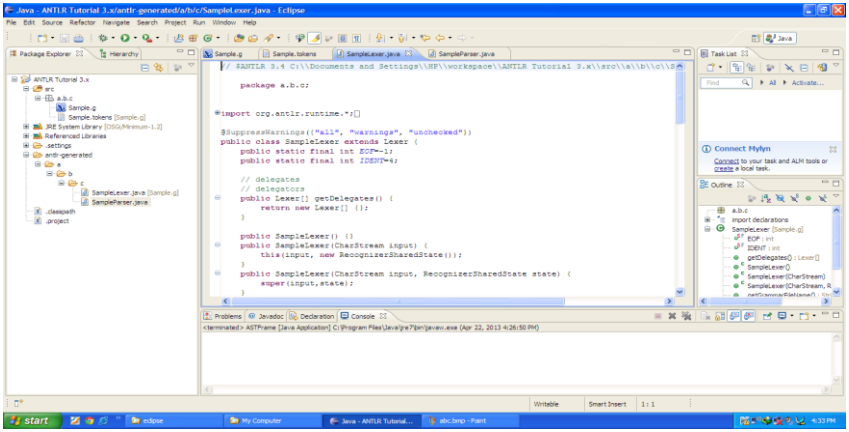
Thử nghiệm 1: Thử nghiệm tính tự động tạo trình phân tích từ vựng (lexer) và trình phân tích cú pháp (parsers) của công cụ ANTLR bằng ví dụ đơn giản tạo gói a.b.c.

Đoạn mã cụ thể như sau:

```
grammar Sample;
options {
  language = Java;
}
@header {
   package a.b.c;
}
@lexer:: header {
   package a.b.c;
}
rule: IDENT ;
IDENT : 'a'..'z'+;
```

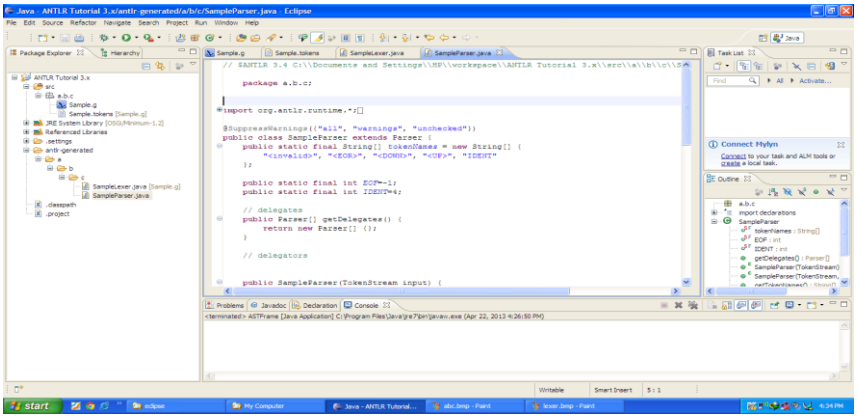
Khi tạo gói a.b.c thì ANTLR sẽ tự động sinh mã cho các file Sample.tokens, SampleLexer.java, SampleParser.java cụ thể như hình sau:

Tự động sinh trình phân tích từ vựng (Lexer)



Hình 3.1: *Tự động trình phân tích từ vựng (Lexer)*

Tự động sinh trình phân tích cú pháp (Parser)



Hình 3.12: *Tự động trình phân tích cú pháp (Parser)*

Thử nghiệm 2: Thử nghiệm tính tự sinh mã của ANTLR.

Cũng như ví dụ trên, chúng ta tiến hành thêm biến x vào trong đoạn mã của tập tin Sample.g

```
grammar Sample;
options {
  language = Java;
}
@header {
   package a.b.c;
}
@lexer:: header {
   package a.b.c;
}
rule: IDENT {x};
IDENT : 'a'..'z'+;
```

ANTLR sẽ tự động sinh mã trong tập tin SampleParser.java thể hiện như sau. Đoạn mã trong tập tin SampleParser.java khi chưa thêm biến x:

```
   public      final      void      rule()      throws
RecognitionException {
            try {

match(input,IDENT,FOLLOW_IDENT_in_rule38);
            }
            catch (RecognitionException re) {
                reportError(re);
                recover(input,re);
            }
            finally {
             // do for sure before leaving
            }
            return ;
       }
```

Đoạn mã trong tập tin SampleParser.java khi đã thêm biến x:

```
   public      final      void      rule()      throws
RecognitionException {
            try {
                {
match(input,IDENT,FOLLOW_IDENT_in_rule38);
                x
                }
            }
            catch (RecognitionException re) {
                reportError(re);
                recover(input,re);
            }
            finally {
             // do for sure before leaving
            }
            return ;
       }
```

KẾT LUẬN

Trong thời đại hội nhập kinh tế thế giới hiện nay, sự trao đổi giữa các công ty dù lớn hay nhỏ với nhau phần lớn đều sử dụng nhiều ngôn ngữ khác nhau. Vấn đề trao đổi thông tin giữa các dân tộc, giữa nhiều cộng đồng trên thế giới với nhau luôn gặp phải những khó khăn, những trở ngại do việc qui định cách viết và hiểu về các thông điệp, số liệu là rất khác nhau.

Phần mềm đa ngữ (Multilingual Software) là phần mềm cho phép người sử dụng chọn lựa ngôn ngữ (ví dụ: tiếng Anh, tiếng Pháp, tiếng Việt...) khi làm việc với phần mềm đó. Đa dạng hóa ngôn ngữ trong các sản phẩm phần mềm là xu hướng của các nhà sản xuất phần mềm hiện nay, bởi tính toàn cầu hóa cũng như phương hướng mở rộng thị trường ra quốc tế. Lựa chọn đối tác dịch chuyên nghiệp là phương pháp mà các nhà sản xuất phần mềm sử dụng.

Ứng dụng ANTLR để xử lý các thông điệp trong đã ngữ hóa phần mềm là một cách để tăng hiệu quả việc toàn cầu hóa và bản địa hóa phần mềm thay vì phải viết một mã nguồn riêng cho mỗi ngôn ngữ.

Qua quá trình tìm hiểu và nghiên cứu ứng dụng ANTLR trong đa ngữ hóa phần mềm đã cho thấy kết quả tương đối tốt. Công cụ này đã cho thấy những ưu điểm, sự tiện lợi, có khả năng ứng dụng trong thực tiễn ở Việt Nam.

Như vậy, sau một thời gian tiến hành nghiên cứu, tác giả đã cơ bản hoàn thành các nội dung mà đề cương đề tài đã đạt ra. Sản phẩm đề tài này có thể được ứng dụng tốt trong việc tìm hiểu, nghiên cứu và triển khai trong lập trình ứng dụng đặc biệt là trong dạy và học ở bậc phổ thông và đại học.