Scrawl

مستندات فنی

پروژه درس اصول طراحی کامپایلر

دانشگاه علم و صنعت ایران
دکتر پارسا

حمید فیض‌آبادی ۹۲۵۲۲۱۷۷
بردیا حیدری‌نژاد ۹۲۵۲۱۱۱۴

پاییز ۹۴

فهرست

چکیده

هدف از این پروژه طراحی یک کامپایلر است تا کار های تحلیل لغوی و تحیلی نحوی یک کد ساختگی را انجام دهد و سپس در انتها یک کد میانی تولید کند.

معرفی پروژه

در این پروژه ما قصد ساختن یک زبان سطح بالا را داریم. این زبان یک زبان خاص منظوره است که برای Web Scraping نوشته می‌شود. Web scraping به عمل استخراج اطلاعات از یک وبسایت گفته می‌شود. فرض کنید یک وبسایت اطلاعات زیادی را روی سایت خود قرار داده است و شما می‌خواهید یک برنامه بنویسید که آن اطلاعات را دریافت و به صورت ساختار یافته ذخیره کنید.
ما اسم زبان طراحی شده را Scrawl گذاشته ایم. زبان Scrawl دارای دستور زبان خاص خود می‌باشد که کد های طولانی استخراج از وب را بسیار کوتاه می‌کند.

تحلیلگر لغوی

در این بخش زبان منظم استفاده شده برای ساختن توکن‌ها برای تحلیلگر نحوی بیان‌شده است:

```
WS  :  ( ' '
       | '\t'
       | '\r'
       | '\n'
       )
    ;

GET             :       'get';
POST  :         'post';
PUT             :       'put';
DELETE          :       'delete';
TEXT  :         'text';

THIS    :       'this';
IF              :       'if';
ELSE  :         'else';

STRING : '"' (ESC | ~('\\'|'"'))* '"';
protected ESC : '\\' ('n' | 'r');

ID : SS (SS | '0'..'9')* ;
fragment SS : 'a'..'z' | 'A'..'Z' | '_' ;

LBR :  '(' ;
RBR :  ')' ;
PLS :  '+' ;
MNS :  '-' ;
MLP :  '*' ;
DIV :  '/' ;
```

```
PWR :  '^' ;

LSS :  '<'  ;
LSQ :  '<=' ;
GRT :  '>'  ;
GRQ :  '>=' ;
EQL :  '==' ;
NEQ :  '!=' ;
AND :  '&&' ;
OR  :  '||' ;
NOT :  '!'  ;


INTEGER: DIGIT+;
fragment DIGIT          :          '0'..'9';

FLOAT : INTEGER '.' INTEGER* EXP? | '.' INTEGER EXP? | INTEGER EXP;
fragment EXP : ('e'|'E') (PLS | MNS)? INTEGER;

COMMENT     : '/*' ( options {greedy=false;} : . )* '*/';

LINE_COMMENT: '//' ~('\n'|'\r')* '\r'? '\n';
```

تحلیلگر نحوی

در این قسمت پارسر نوشته شده برای بررسی متن برنامه را بیان شده است.

```
root  returns [String code]:
   mainRoutine (procedure)*;

procedure returns [String code]:
   'procedure' ID block ;

mainRoutine returns [String code]:
   'main' block ;

block returns [String code]:
   '{' (statement)* '}';

statement returns [String code]:
   reqSt
   |assSt
   |foreachSt
   |parseSt
   |printSt
   |ifSt ;

reqSt returns [String code] :
   getReqSt|postReqSt ;
```

```
getReqSt returns [String code]:
   GET exp block;

postReqSt returns [String code]:
   POST exp block;

ifSt returns [String code]:
   (IF conditionSt a=block)(ELSE b=block)?;

conditionSt returns [String code]:
   a=exp EQL b=exp;

assSt returns[String code]:
   ID '=' exp ';';

foreachSt returns[String code]:
   'foreach' exp block;

parseSt:
   'parse' 'first' exp 'by' ID ';'|'parse' 'last' exp 'by' ID ';';

printSt returns [String code]:
   'print' exp ';';

exp returns [String code, Type type]:
   x=multExpr (PLS x=multExpr|MNS x=multExpr)*;


multExpr returns [String code, Type type]:
   x=atom (MLP x=atom)*;

atom returns [String code, Type type]:
   ID|STRING|THIS('@'(TEXT))?|integer|selector( index('@'(TEXT|ID))?)?;

index returns[Integer value]:
   '[' integer ']';

selector returns[String code, Type type]:
   LBR exp RBR;


integer returns[Integer value]:
   INTEGER ;
```

ابزار طراحی کامپایلر

ANTLR که خلاصه شده ANother Tool for Language Recognition یک ابزار تولید کننده مترجم و پارسر است که ما برای طراحی کامپایلر از آن استفاده کردیم و به ما اجازه می دهد که گرامرهای زبان در قالب ANTLR syntax که چیزی شبیه EBNF)Extended Backus-Naur Form) است و یا AST syntax را تعریف کنیم. میدانیم که اولین فاز کامپایلر lexical analysis است که بر روی کاراکترهای ورودی عمل میکند و دومین مرحله parsing است که بر روی توکن های بدست آمده از فاز اول کار میکند. این دو فاز از بخشهای اساسی کامپایلر هستند. ANTLR بصورت اتوماتیک lexical analyzer و parser را برای گرامری که ما برای آن فراهم میکنیم می سازد. ANTLR چیزی بیش از یک زبان تعریف گرامر است و ابزاری است که اجازه پیاده سازی گرامر را با تولید اتوماتیک lexer و parser (و tree parser) به زبانهای جاوا، C++ و یا Sather میدهد. از آنجایی که ANTLR از مکانیسم تشخیص یکسانی برای لغت یابی، تجزیه و تجزیه درخت استفاده میکند، lexer هایی بسیار قویتر از lexer هایی که مبتنی بر DFA هستند (مانند آنهایی که lex تولید میکند) تولید میکند. ANTLR ، lexer های)LL(k تولید میکند. ANTLR یک predicated LL(k) parser است که بصورت open source و رایگان ارائه شده است. برخ ف تجزیه کننده و لغت یاب های ک سیک نظیر پارسرهای /yacc bison و یا پویشگرهای ANTLR ،lex/flex روش جدیدی برای ترجمه زبان ارائه داده است چونکه مابین lexer و parser تمایز کمی قائل شده و از تعریف گرامریک EBNF یکسانی برای هر دو استفاده می کند. این ویژگی باعث شده ما برای هر دو ابزار (parser و lexer) گرامر واحدی تعریف کنیم و سرعت تولید مترجم را با ببریم.

تولید کد میانی

در این پروژه برای تولید کد میانی از اسمبلی بایت کد جاوا استفاده میکنیم سپس با استفاده از اسمبلر Jasmin تبدیل به بایت کد JVM (فایل با پسوند .class) مي‌شود. JVM فایل با پسوند .class را گرفته و آن را اجرا مي‌كند. با توجه به ويژگي‌هاي JVM در هر فایل .class يك كلاس با همان نام فایل وجود دارد. دستورات stack based، JVM مي‌باشند. يعني اينكه براي انجام همه اعمال از پشته استفاده مي‌كنيم. هر تابع در JVM داراي يك پشته تحت عنوان operand stack مي‌باشد و همچنين هر تابع داراي متغير هاي محلی مي‌باشد كه به صورت شماره گذاری شده ميباشند (براي ارجاع به آنها از شماره آنها استفاده مdكنيم) همیشه متغیر شماره 0 حاوی مقدار اشارهگرthis (اشارهگر به شی فعلی) می‌باشد.

آموزش برنامه نویسی با کامپایلر

انواع داده ای

عدد صحیح: این متغیر در خود یک عدد را جای می‌دهد. اینکه این عدد چند بایت اشغال می‌کند بستگی به مقدار عدد دارد برای مقال a = 120 یک متغیر یک بایتی ایجاد می‌کند.

متن: رشته یا متن دنباله ای از کارکتر ها است و از آنجایی که پروتکل HTTP (Hypertext Transfer Protocol) بر پایه متن ساخته شده است نوع داده‌ای String یکی از مهم ترین انواع داده‌ای به حساب می‌رود.

داکیومنت: نتیجه گرفته شدن یک وب پیج یک داکیومنت است.

المنت: یک تگ در زبان html را یک المنت می‌گویند. این المنت ها هستند که حاوی اطلاعات قابل استخراج می‌باشند.

المنت ها: به لیستی از المنت ها که معمولا نتیجه خروجی یک سلکت است می‌گویند. روی المنت ها میتوان پیمایش انجام داد و آنها itrative هستند.

اشاره گر this: در زبان scrawl وقتی وارد یک بلاک دستوری مانند بلاک get می‌شویم به متغییری به نام this دسترسی داریم. این متغیر به مقدار به دست آمده توسط بلاک اشاره می‌کند.

بدنه اصلی

هر برنامه در این زبان از یک روتین اصلی ساخته شده است. برای نمایش این روتین از کلمه کلیدی main استفاده می‌کنیم.
مثال

```
main {
  // you statements
}
```

در زبان scrawl برای نمایش یک بلاک از {} استفاده می‌شود.
بدنه تابع main از چند جمله دستوری یا statement ساخته شده است که این دستورات یه صورت ترتیبی اجرا می‌شوند.
گرفتن صفحه وب: مهم ترین کاری که برای یک برنامه web scraping وجود دارد گرفتن یک صفحه وب است. با اجرای این دستور یک system-call رخ می‌دهد تا دستگاه یک رکوئست get در پروتوکل http به سایت مورد نظر بفرست و جواب را بگیرد. این دستور از نوع blocking است. یعنی برنامه متوقف می‌شود تا جواب سایت برگردد. این بلاک روی متغیر this تاثیر میگذارد و آنرا مساوی داکیومنت نتیجه قرار می‌دهد.

```
get “https://google.com”{
  // your statements
}
```

دستور انتصابی: یکی از مهمترین دستورات برنامه نویسی دستور انتصاب است. زبان scrawl در هنگام مقدار دهی متغیر را می‌سازد و نیازی به تعریف متغیر از قبل وجود ندارد.

```
a = 3;
b = “Hello”;
```

شرط: زبان scrawl از دستور های شرطی پشتیبانی می‌کند که برای چک کردن تساوی دو عبارت به کار می‌رود.

```
if(b==”Hello”)
{
  // if equal statemesnts
}
else
{
  //if not equal statements
}
```

دستور برای هر: اگر ما لیستی از المنت هارا در اختیار داریم و می‌خواهیم برای هر المنت کاری انجام بدهیم از دستور برای هر استفاده می‌کنیم. این دستور متغیر this را عوض می‌کند و برابر المنت فعلی قرار می‌دهد.

```
foreach elements{
  // do something
}
```

دستور چاپ: بعد از گرفتن اطلاعات ما باید آنهارا چاپ کنیم. برای این عمل از دستور print استفاده می‌کنیم. در جلوی دستور print یک عبارت از نوی string یا integer قرار میگیرد.

```
a = 5;
print a;
b = “hello”
print b;
```

عملگر سلکت: برای به دست آوردن یک المنت در یک صفحه html زبان scrawl از زبان کوئری css selector پشتیبانی میکند. خروجی ایت عمگلر لیست المنت ها است. این دستور به صورت پیش فرش روی اشاره گر this اجرا می‌شود.
عملگر attribute این عملگر برای گرفتن اطلاعات از یک المنت استفاده می‌شود و خروجی آن string است. مثال:

```
main{
  get “https://google.com”{
    foreach(“//li”){
      print this@text;
    }
```

```
  }
}
```

در مثال بالا صفحه گوگل گرفته میشود و به ازای هر تگ li متن داخلی آن چاپ می‌شود.

سورس برنامه

- Scrawl.g : این فایل سورس برنامه‌ی کامپایلر کامپایلر antlr می‌باشد و شامل گرامر زبان و تحلیلگر لغوی مباشد. همچنین گرامر ویژه نیز در این فایل پیاده سازی شده است.
- Scrawl.jar : این فایل برنامه‌ی کامپیالر نوشته شده می‌باشد که به وسیله‌ی ماشین مجازی جاوا قابل اجرا است.
- examples: این پوشه حاوی فایل‌های و برنامه‌های نوشته شده برای این کامپایلر می‌باشد.

منابع

- Java virtual machine ,Jon Meyer &Troy Downing ,O'Reilly(1997)

1. http://artima.com/jvm/index.html
2. https://en.wikipedia.org/wiki/Java_bytecode_instruction_listings
3. http://jasmin.sourceforge.net/instructions.html
4. http://classfileanalyzer.javaseiten.de/
5. Aho, Sethi, Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986. ISBN 0-201-10088-6