اصول طراحی کامپایلر

درس ۱۵

تحلیل نحوی (۱۰)

تولید خودکار تجزیه‌گر

Syntax Analysis (10)

Automatic Parser Generation

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/compiler

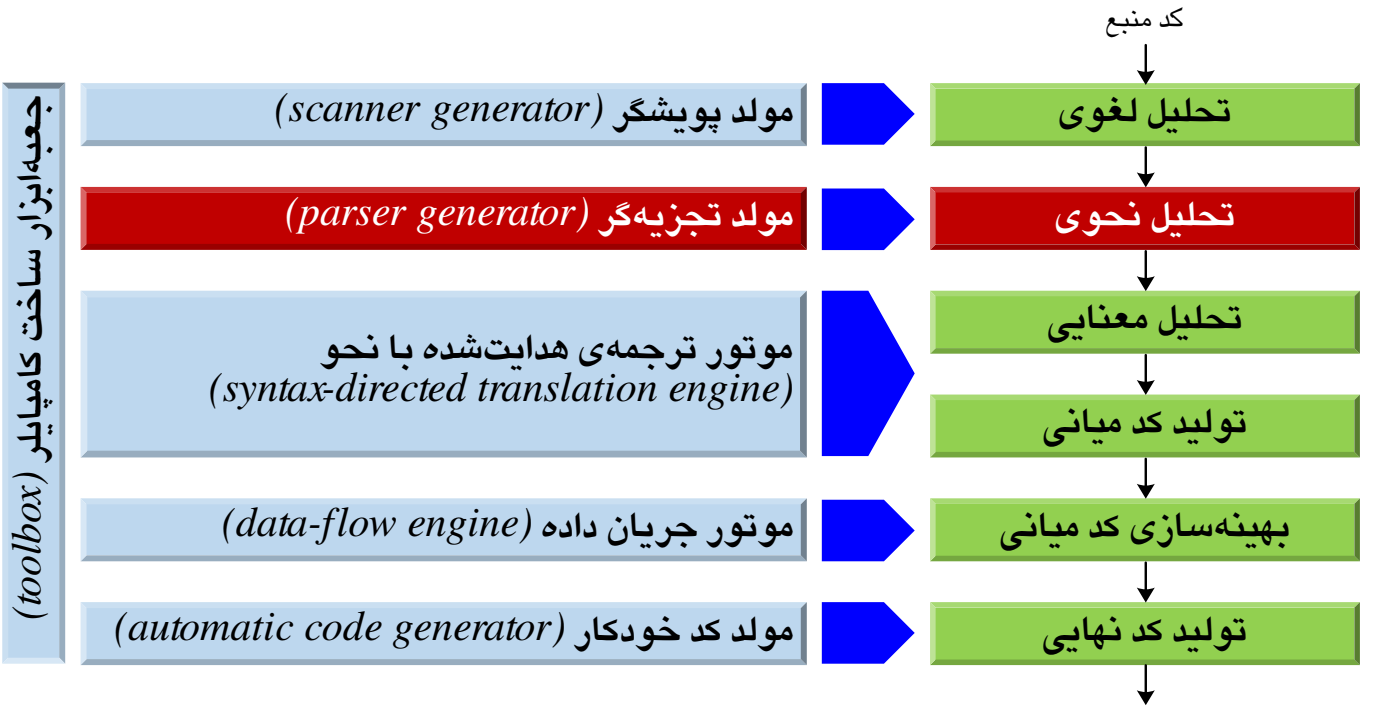
اصول طراحی کامپایلر

تحلیل نحوی
تولید خودکار تجزیه‌گر

۱

مقدمه

تولید خودکار تحلیل‌گر نحوی (تجزیه‌گر/پارسر)



تولید خودکار تحلیل‌گر نحوی (تجزیه‌گر/پارسر)

نام مولد	زبان برنامه‌سازی	سیستم	تجزیه‌گر
YACC	C	UNIX	پایین به بالا: LALR(1)
BISON	C	UNIX, LINUX, Windows	پایین به بالا: LALR(1)
ANTLR	JAVA	UNIX, LINUX, Windows	بالا به پایین: $LL(k)$
JavaCC	JAVA	JVM	بالا به پایین: $LL(k)$

اصول طراحی کامپایلر

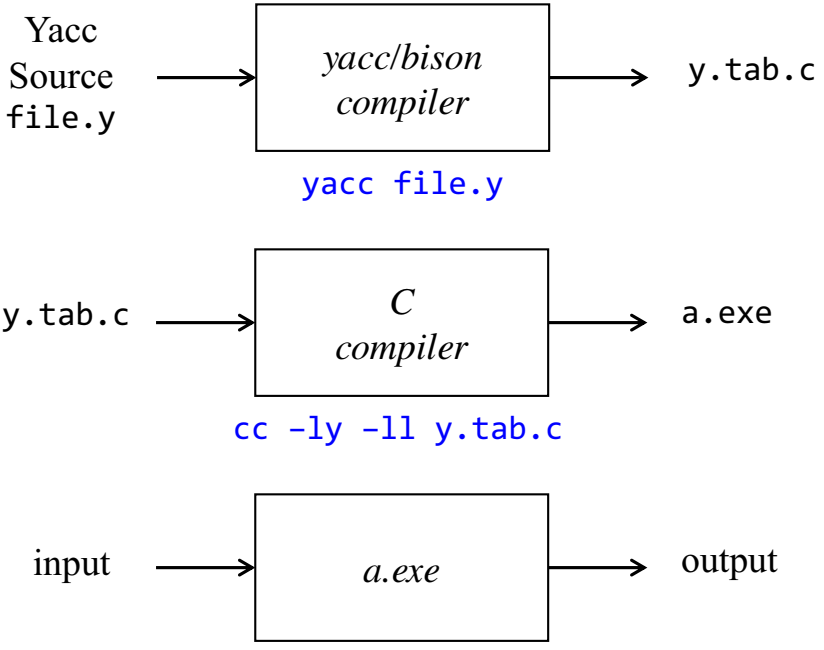
تحلیل نحوی
تولید خودکار تجزیه‌گر

۲

YACC

YACC

Yet Another Compiler Compiler



YACC

قالب کد منبع

```
[Declarations]

%%

[Translation Rules]

%%

[Auxiliary Procedures]
```

اعلان‌ها:
تعریف توکن‌ها و خصوصیات آنها

قواعد ترجمه:
قواعد گرامر و
کنش‌هایی که در مواجهه با آنها باید انجام شود
(در قالب کد C)

روال‌های کمکی:
سایر توابع لازم و مورد نیاز
(در قالب کد C)

YACC

ارتباط با اسکنر

yylex()	تابع اسکنر

YACC

برخورد با تداخل‌های جدول تجزیه

تداخل شیفت ـ کاهش *Shift-Reduce Conflict*	**انتخاب شیفت** (مطابقت با رشته‌ی طولانی‌تر)
تداخل کاهش ـ کاهش *Reduce-Reduce Conflict*	**انتخاب آنچه در لیست زودتر آمده است.**

YACC

برخورد با خطا

برخورد با خطا
Error Handling

استفاده از قواعد تولید برای ادراه‌ی خطا

مثلاً: `lines: error '\n' {...}`

وقتی به یک خطا برخورد می‌کنیم، تا انتهای خط نادیده گرفته می‌شود.

نام	کاربرد
`error`	توکن ویژه‌ی خطا
`Yyerror(string)`	روال از پیش تعریف‌شده برای چاپ پیغام‌های خطا
`yyerrok()`	reset کردن پرچم‌های خطا (error flags)

YACC

مثال (۱ از ۳)

```
%{
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#define YYSTYPE int /* integer type for YACC stack */

%}

%token NUMBER
%left '+' '-'
%left '*' '/'
%left UMINUS

%%
```

YACC

مثال (۲ از ۳)

```
lines   :  lines expr '\n'        {printf("%d\n", $2);}
        |  lines '\n'
        |  /* empty, i.e., epsilon */
        |  lines error '\n' { yyerror("Please reenter:"); yyerrok; }
        ;
expr    :  expr '+' expr     { $$ = $1 + $3; }
        |  expr '-' expr     { $$ = $1 - $3; }
        |  expr '*' expr     { $$ = $1 * $3; }
        |  expr '/' expr     { $$ = $1 / $3; }
        |  '(' expr ')'      { $$ = $2; }
        |  '-' expr  %prec UMINUS  { $$ = - $2; }
        |  NUMBER       { $$ = atoi(yytext);}
        ;

%%
#include "lex.yy.c"
```

YACC

مثال (۳ از ۳): فایل LEX

```
%{
%}
Digit        [0-9]
IntLit       {Digit}+
%%
[ \t] {/* skip white spaces */}
[\n] {return('\n');}
{IntLit}                              {return(NUMBER);}
"+"                                   {return('+');}
"-"                                   {return('-');}
"*"                                   {return('*');}
"/"                                   {return('/');}
.          {printf("error token <%s>\n",yytext); return(ERROR);}
%%
```

file.lex

YACC

تعیین شرکت‌پذیری و تقدم

تقدم *Precedence*	**به ترتیب لیست، افزایش تقدم** عملگرهای واقع در یک خط مشترک، هم‌تقدم هستند.
شرکت‌پذیری *Associativity*	**با نشانه‌گذاری: چپ/راست/بدون شرکت‌پذیری** (مثلاً عملگر ضرب نقطه‌ای بردارها، شرکت‌پذیری ندارد.)

```
%left  '+' '-'
%left  '*' '/'
%left  UMINUS
%right '^'
```

YACC

قواعد معنایی

کنش‌های معنایی
Semantic Actions

به هر قاعده‌ی تولید یک مجموعه دستور مرتبط می‌شود که در صورت استفاده از آن قاعده اجرا می‌شود.

هر عنصر در قاعده‌ی تولید، با یک مقدار مرتبط می‌شود.

نام متغیر	کاربرد
YYSTEP	نوع مقدار بازگشتی (آنچه در پشته قرار می‌گیرد.)
$$	مقدار بازگشتی در صورت کاهش با قاعده‌ی فعلی
$$i$$	مقدار بازگشتی iامین عنصر در قاعده

کنش‌ها می‌توانند در داخل یک قاعده‌ی تولید درج شوند (با این کنش‌ها مانند یک ناپایانه برخورد می‌شود).

```
expr    :  expr { $$ = 32;}  '+' expr    { $$ = $1 + $2 + $4; };

is equivalent to

expr  :  expr $ACT '+' expr {$$ = $1 + $2 + $4;};
$ACT : {$$ = 32;};
```

YACC

سبک برنامه‌نویسی

۱ از کنش‌های داخل قاعده‌ی تولید اجتناب کنید!

آنها را با علامت‌گذارها (markers) جایگزین کنید.

۲ سمت راست قواعد تولید را کوتاه نگه دارید!

بهتر است کمتر از ۴ نماد باشد.

۳ از کلمات رزروشده‌ی زبان C اجتناب کنید!

به قواعد زبان C دقت کنید!

۴ سعی کنید برای هر قاعده‌ی تولید نمادهای منحصربه‌فردی بیابید!

برای مثال:

$$array \rightarrow aelist\ ]$$
$$aelist \rightarrow aelist, \mathbf{id} \mid ahead$$
$$ahead \rightarrow \mathbf{id}\ [\ \mathbf{id}$$

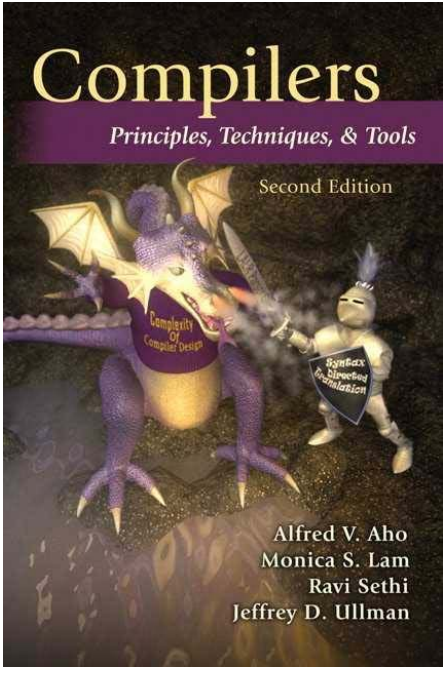
اصول طراحی کامپایلر

تحلیل نحوی
تولید خودکار تجزیه‌گر

۳

منابع

منبع اصلی



A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman,
Compilers: Principles, Techniques and Tools,
Second Edition, Addison-Wesley, 2007.

Chapter 4 (4.9)