اصول طراحی کامپایلر

گروه کامپیوتر دانشگاه آزاد اسلامی واحد زنجان

مدرس :مهدی بازرگانی

آخرین بروز رسانی :پاییز ۱۳۹۳

فصل سوم

تحلیل گرلغوی

تحلیل گرلغوی(مثال)

درشکل بالامشاهده میشودویرایشگردرمکانهایی که برنامه نویس کلیدEnterزده است, کاراکترکنترلی /n وبرای نشان دادن آخرفایل کاراکتر کنترلی /z رادرج کرده است.

تحلیل گرلغوی برناه مبدأرابصورت جریانی از کاراکترها میخواند ولغات را استخراج میکندازآنجایی که L.A متن ورودی راپویش یا scanمیکندبر L.A پویشگریاscanner نیزگفته میشود.

تحلیل گرلغوی(مثال)

به مثال زیرتوجه کنید:

```
Program sample;
Var  i=integer;
Begin
i:=i+2;
Write ln(i);
End.
```

جداکننده

[p	r	o	g	r	a	M]		[s	a	m	p	l	E]	;	
/n	v	a	r		i	:	i	n	t	e	g	e	r	;	
		B	e	g	i	n		i	:	=	i	+	2	;	/n
w	r	i	t	e	l	n	(	i	)	;	/n	e	n	d	.
/z															

یک نمونه برنامه جهت بیان تفاوتها

token	Lexme	Pattern
Const	Const	Const
If	If	If
Relation	< ,> , ≤ ,...	< or > or ...
Id	Pi	با حروف الفبا شروع و به دنبال آن حرف یا رقم می آید
Num	3.1416	هر ثابت عددی
literal	" Test"	هررشته نویسه ای که بین دو علامت "" قرار گیرد

Zau-93-94-2

تعاریف

❖ (نشانه) Token:

دنباله ای از کاراکترها در ورودی که با یک ارتباط منطقی با هم دارند، توکن یا نشانه را تولید می کنند.

❖ (لغت):Lexeme

دنباله ای از کاراکترها در ورودی که با یک الگو برای یک توکن منطبق شده است. آنچه در برنامه ما ظاهر می شود. Lexeme ها معمولاً مطابق با الگویی هستند که برای یک نشانه در نظر گرفته می شوند.

❖ Pattern(الگو):

به شکل کلی که یک توکن می تواند داشته باشد الگوی آن توکن می گویند . به عبارت دیگر رشته ها و عبارتهای مختلفی در ورودی وجود دارد که برای همه آنها توکن یکسانی تشخیص داده می شود ، به شکل کلی این رشته ها الگوی آن توکن می گوییم .

Zau-93-94-2

تکنیکهایی برای بالابردن کارآیی اسکنر

۱-تکنیک استفاده از بافرورودی(Input Buffering)

–استفاده از یک بافر به طول 2N کاراکتر

۲-تکنیک استفاده از نگهبانها (Sentinels)

–حالت توسعه یافته بافر ورودی برای اصلاح مشکلات آن

Zau-93-94-2

خطاهای واژه ای(Lexical Error)

❖ خطاهای بسیار کمی را درفاز تحلیل لغوی می توان تشخیص داد.

❖ تحلیل گر لغوی دید وسیعی نسبت به ورودی ندارد ومعمولا چند کاراکتر جلو و عقب کاراکتر جاری را می تواند ببیند .

❖ در صورتی که پس ازتشخیص یک واژه توسط تحلیل گر لغوی این واژه توسط هیچ یک از الگوهای موجود در زبان قابل تولید نباشد یک خطای لغوی رخ داده است .

✓ Var Temp: integer;

✓ Temp#:= 1o;

✓ T=123.423.34;

❖ مواردی هم پیش خواهد آمد که تحلیل گر لغوی هیچ کاری نمی تواند انجام دهد. در اینگونه موارد خطاپرداز فراخوانی می شود تا به نحو مقتضی خطا را مدیریت نماید.

Zau-93-94-2

الگوریتم بافر ورودی(input Buffering)

- دراین روش بافری به طول دوبخشN کاراکتری در نظر گرفته می شود، بطوریکه Nبرابر تعداد کاراکترهایی است که برروی هر بلاک حافظه قابلیت ذخیره سازی دارد.
- تفاوتی که در این وجود دارد با هربار عمل خواندن به جای یک کاراکتر، N کاراکترازحافظه با یک دستور خوانده می شود ودر بخش جای خالی بافر قرار می گیرد .

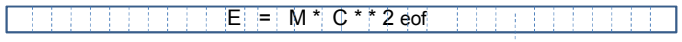
الگوریتم بافر ورودی(input Buffering)

❖این شیوه برای تشخیص توکن هایی استفاده می شود که عمل تشخیص آنها نیاز به پیش نگری دررشته ورودی دارد.

❖ مانند: if A<=B

Input buffering

- Sometimes lexical analyzer needs to look ahead some symbols to decide about the token to return
  - In C language: we need to look after -, = or < to decide what token to return
  - In Fortran: DO 5 I = 1.25
- We need to introduce a two buffer scheme to handle large look-aheads safely



الگوریتم بافر ورودی(input Buffering)

- چنانچه تعداد کاراکترهای خوانده شده کمتراز Nباشد، یک کاراکترخاص(eof)به معنای انتهای ورودی قرار می گیرد.
- دراین روش ازدو اشاره گر استفاده می شود. (Lexeme Beginningو Forward). رشته ای از کاراکترها که بین دو اشاره گر قرار می گیرد، لغت جاری خواهد بود.
- در ابتدا هردو اشاره گر به اولین کاراکتر از لغت بعدی اشاره می کنند، سپس اشاره گر پیشرو(forward) در صورتیکه به کاراکتر مورد نظر برسد،یک مکان به جلو حرکت می نماید تااینکه بتواند طبق الگوزیررشته ای ازورودی راجدا نماید.
- همینکه لغت بعدی تعیین شد forward به سمت راست ترین کاراکترآن لغت اشاره می کند. پس ازاینکه لغت پردازش گردید،هردواشاره گر به سمت راست ترین کاراکترآن لغت اشاره خواهندکرد .

الگوریتم بافر ورودی(input Buffering)

❖ نکته) این روش دراغلب موارد خوب عمل می کند، اما میزان پیش نگری درآن محدود به اندازه بافرهاست،بااین محدودیت ممکن است که تشخیص توکن ها میسرنباشد، زیرا فاصله ای کهforward بایدبپیماید بیشترازطول بافراست. نهایت پیشروی forward به اندازه 2Nکاراکترخواهد بود .

الگوریتم بافر ورودی(input Buffering)

```
if forward at the end of first half then
  Begin
      Reload second half
      forward:=forward+1
  End;
Else if forward at the endof second half then
  Begin
      Reload first half
      move forward to Beginning of first half
  End;
Else forward:= forward+1;
```


Sentinels(C Code)

```
Switch (*forward++) {
    case eof:
        if (forward is at end of first buffer) {
            reload second buffer;
            forward = beginning of second buffer;
        }
        else if {forward is at end of second buffer) {
            reload first buffer;\
            forward = beginning of first buffer;
        }
        else /* eof within a buffer marks the end of input */
            terminate lexical analysis;
        break;
    cases for the other characters;
}
```

E = M eof * C * * 2 eof eof

الگوریتم بهبود یافته بافر ورودی(استفاده از نگهبانها)

❖ برای هربارجلو رفتن اشاره گرforward دوعمل مقایسه انجام میشود بااندکی اصلاحات دردو قسمت بافروالگوریتم اجرای عمل میتوان تعداداین مقایسه ها را کاهش داد.

❖ برای کاهش عمل مقایسه کافیست ازیک کاراکترنگهبان(sentinel)همانند eof که کاراکتر ویژه ایست وامکان حضوردررشته ورودی را ندارد،استفاده نماییم. دراینجا درانتهای هریک از بخش های بافریک کاراکترeof قرار می دهیم .

نکات الگوریتم استفاده از نگهبانها

- دراین حالت دربیشتر موارد برای کنترل اینکه آیا اشاره گر پیشرو به eof رسیده یا خیر،به یک آزمون نیاز است، فقط هنگامیکه به انتهای نیمه بافریابه انتهای فایل ورودی می رسد، آزمون های بیشتری صورت می گیرد. چونN کاراکتر ورودی بین eofها قراردارند، متوسط تعدادشرطها (آزمون ها) برای هر کاراکترورودی بسیارنزدیک به یک می باشد .

- **نکته)** اگر Nرا خیلی کوچک بگیریم،عمل خواندن بیشتر می شود،تابافرپرشود واگرNرابزرگتر ازحدمجاز بگیریم، نیمه اول بافربایکبارReload پرنمی شود.آنوقت الگوریتم اول کارایی ندارد. باید Nباندازه بلاک مطابق باشد.

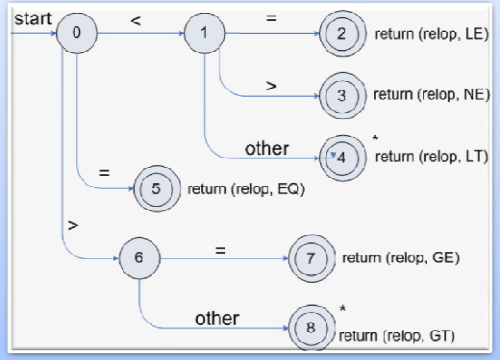
- **نکته)** برای تشخیص الگوی تکن ها دربسیاری از زبانها ازعبارات منظم استفاده می شود. دراین میان برخی ازالگوها توسط گرامرها وعبارات مستقل از متن قابل توصیف هستند .

Sentinels(Pascal Code)

```
forward:= forward+1;
If forward↑=eof then
Begin
      if forward at the endof first half then
    Begin
      Reload second half
      forward:= forward+1;
    End;
  Else if forward at the endof second half then
   Begin
      Reload first half
      forward:= forward+1;
   End;
  Else
     Terminate lexical Analyzer;
End;
```


دیاگرام حالت

- Transition diagram for relop



تشخیص توکن ها در تحلیلگرلغوی

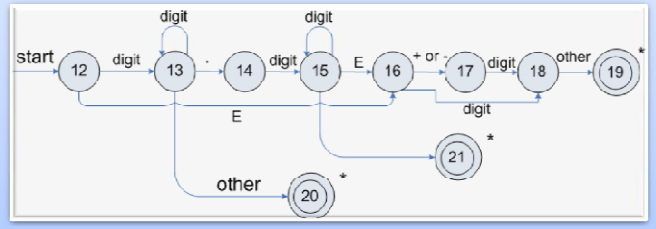
- ابزاراصلی تشخیص توکنها،نمودارهای گذار(Transition Diagram) می باشند .
- برنامه نویسی در تحلیلگر لغوی در ابتدا براساس توکن های موجود ؛نمودارهای گذار؛ برای تشخیص توکنها رسم می شود .
- در اصل نمودار گذار عملی را نشان می دهد که توسط تحلیلگر لغوی برای انجام توکن بعدی انجام می شود .
- نمودار گذار مانندیک ماشین متناهی (FSA) می باشد .مکان ها در این نمودار به صورت دایره ترسیم می شوند و حالت نامیده می شود.
- حالتها توسط خطوط جهتداری به هم وصل می شوند که یال گفته می شوند .

دیاگرام حالت -۲

- Transition diagram for reserved words and identifiers

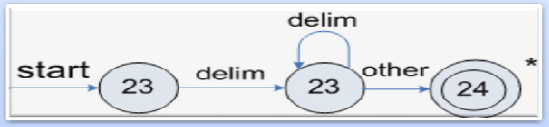
دیاگرام حالت -۳

- Transition diagram for unsigned numbers



دیاگرام حالت - ۴

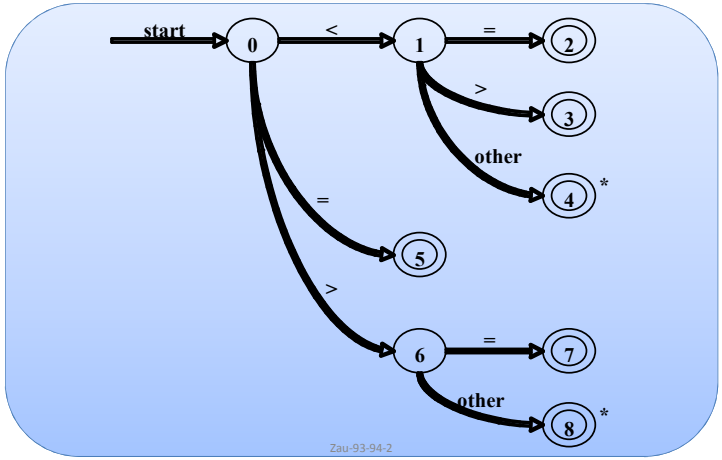
- Transition diagram for whitespace



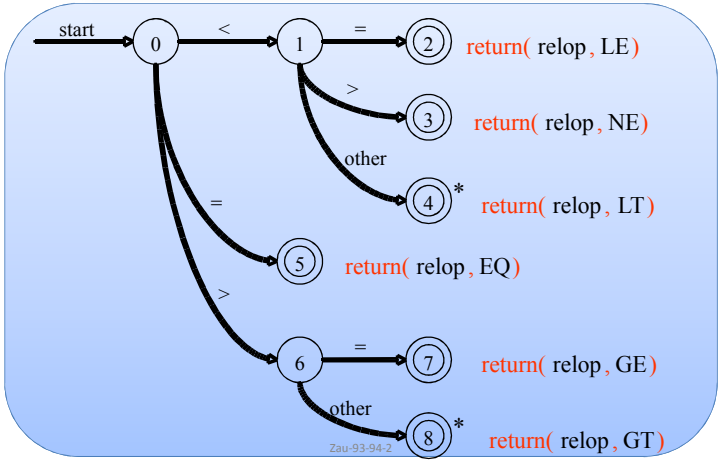
معماری تحلیلگر لغوی برپایه دیاگرام حالت

```
TOKEN getRelop()
{
    TOKEN retToken = new (RELOP)
    while (1) {         /* repeat character processing until a
                                return or failure occurs            */
    switch(state) {
        case 0: c= nextchar();
                if (c == '<') state = 1;
                else if (c == '=') state = 5;
                else if (c == '>') state = 6;
                else fail(); /* lexeme is not a relop */
                break;
        case 1: ...
        ...
        case 8: retract();
                retToken.attribute = GT;
                return(retToken);
    }
```

یک نمودار انتقال



نمودار انتقال



قسمتی از کد اسکنر

```
class Scanner {
   InputStream _in;
   char       _la; // The lookahead character
   char[]     _window; // lexeme window
   Token  nextToken() {
    startLexeme();    // reset window at start
        while(true) {
            switch(_state) {
                case 0: {
                _la = getChar();
                if (_la == '<') _state = 1;
                    else if (_la == '=') _state = 5;
                    else if (_la == '>') _state = 6;
                    else failure(state);
                }break;
                case 6: {
                    _la = getChar();
                    if  (_la == '=') _state = 7;
                    else _state = 8;
                }break;
        }   }   }}
```

```
case 7: {
    return new Token(GEQUAL);
}break;

case 8: {
    pushBack(_la);
    return new Token(GREATER);
}
```

Zau-93-94-2

نکات تحلیلگر لغوی

- کلمات کلیدی مانند شناسه ها تشخیص داده می شوند یعنی ابتدا توسط ماشین مربوط به پذیرنده شناسه ها تشخیص داده شده و سپس با لیست کلمات کلیدی مقایسه می شوند درصورت تطبیق به عنوان کلمه کلیدی وگرنه به عنوان شناسه تشخیص داده می شوند .
- کلمات کلیدی را نیز می توان توسط ماشینهای جداگانه تشخیص داد.
- ماشین زیر کلمه کلیدی while را در زبان پاسکال نشان می دهد :

Zau-93-94-2

دیاگرام حالت

- یکی از ابزارهای پیاده سازی دستی برای اسکنر، دیاگرام انتقال حالت می باشد . یک دیاگرام یک گراف جهت دار است که هریک از گره های آن معرف یک وضعیت می باشد. یکی از وضعیت ها به عنوان وضعیت شروع و یکی یا چند تا از وضعیت ها بعنوان وضعیت خاتمه تلقی می شود. برچسب هایی روی لبه های گراف قرارداده می شود که مشخص می کند در چه صورتی می توانیم از یک وضعیت به وضعیت دیگر برویم . هر دیاگرام انتقال معرف یک زبان می باشد.
- با خواندن کاراکترهای یک رشته و تطبیق آنها با برچسب های دیاگرام انتقال معرف یک زبان و پیمایش آن دیاگرام می توان مشخص نمود که آیا آن رشته متعلق به زبان مورد نظر است یا خیر.

Zau-93-94-2