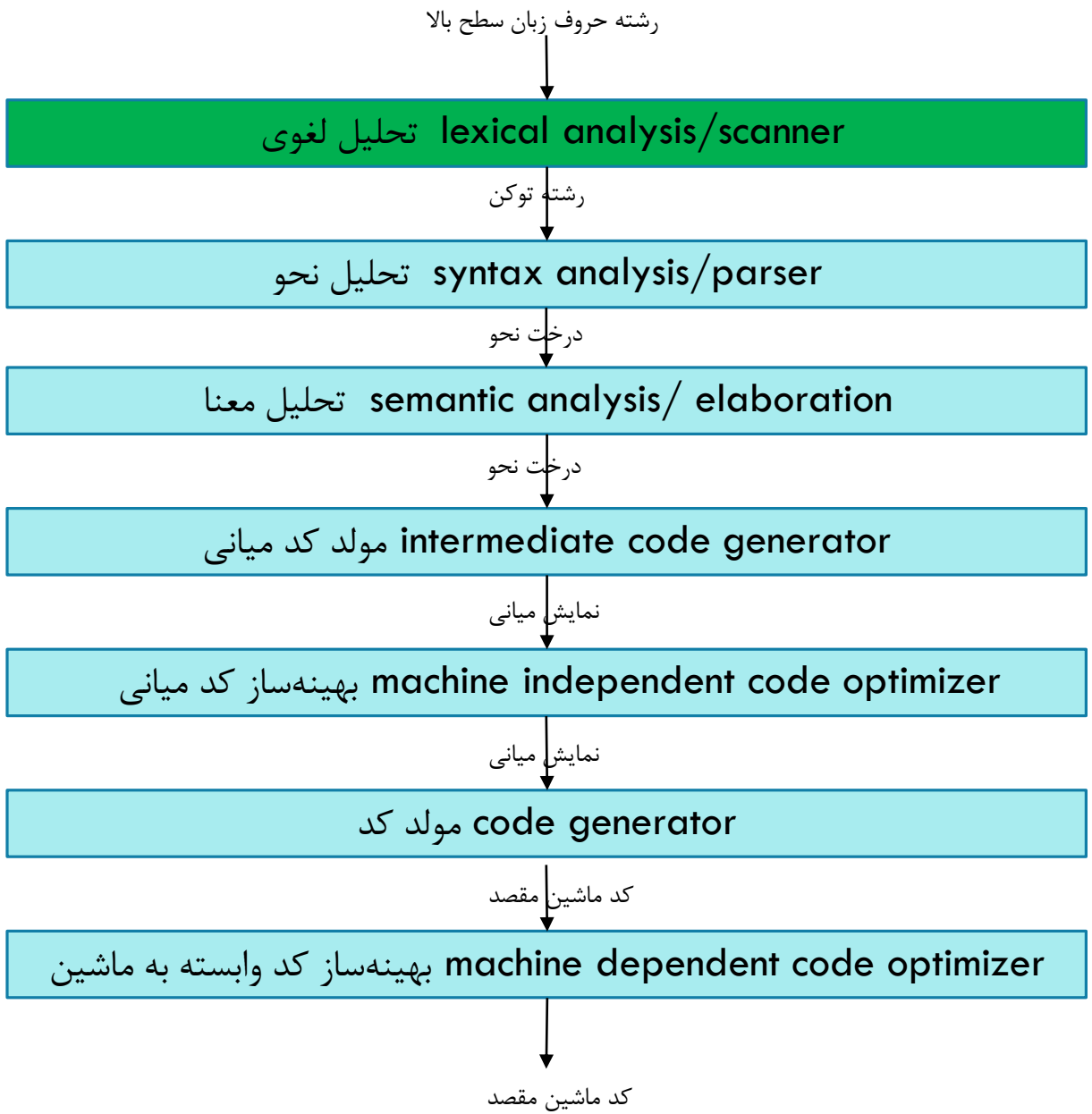
کامپایلر
پویشگر
مختصری درباره پیاده‌سازی

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان


رشته حروف زبان سطح بالا
lexical analysis/scanner تحلیل لغوی
رشته توکن
syntax analysis/parser تحلیل نحو
درخت نحو
semantic analysis/ elaboration تحلیل معنا
درخت نحو
intermediate code generator مولد کد میانی
نمایش میانی
machine independent code optimizer بهینه‌ساز کد میانی
نمایش میانی
code generator مولد کد
کد ماشین مقصد
machine dependent code optimizer بهینه‌ساز کد وابسته به ماشین
کد ماشین مقصد

پیاده‌سازی پویشگر

نظریه زبان‌ها ابزاری خودکار جهت ساخت پویشگر

تفاوت خ‌م‌م و پویشگر

- علاوه بر تشخیص اعتبار تکه، آن را شناسایی نیز می‌کند

سه روش تبدیل خ‌م‌م به کد اجرائی

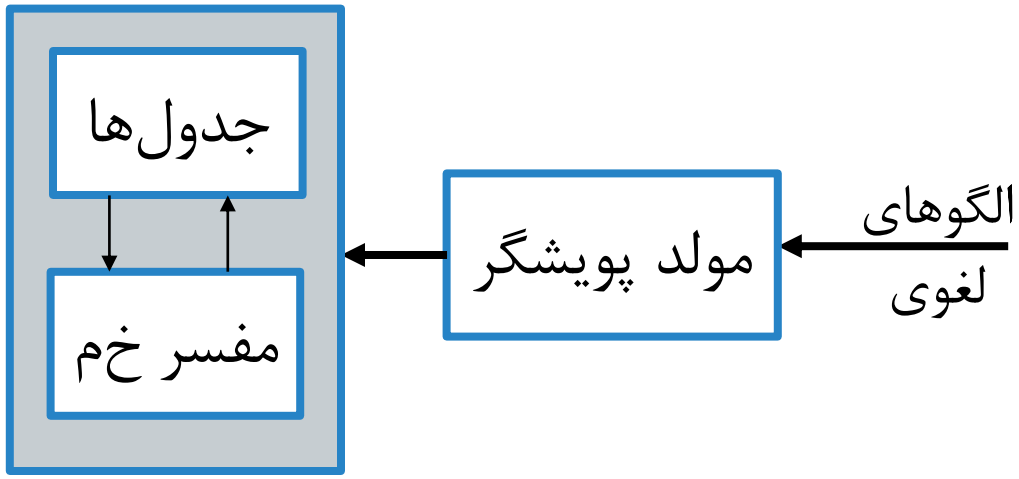
- پویشگر جدول-گرا
- پویشگر مستقیما-کد
- پویشگر دستی

با وجود مرتبه اجرایی یکسان دارای زمان اجرای متفاوت

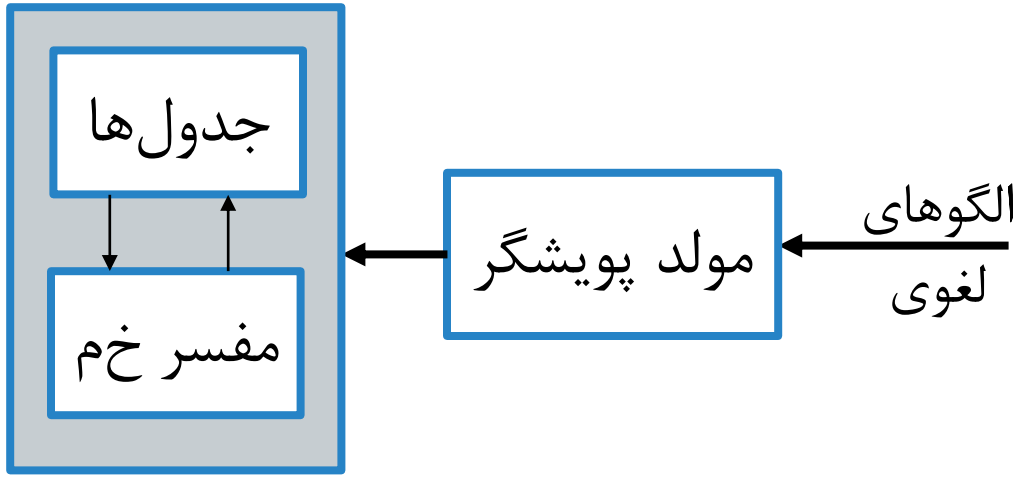
پویشگر جدول-گرا

استفاده از

- پویشگری سبک جهت کنترل
- مجموعه‌ای از جداول تولیدی جهت نگهداری دانش زبان



پویشگر جدول-گرا



نمایش خودکاره در قالب ساختار داده

- جدول انتقال دو بعدی
- هر خانه جدول مشخص‌گر
  - یا انتقال به حالت دیگر
  - یا تحویل تکه
  - یا اعلام خطاء

جدول دیگر

- مشخص‌کننده اینکه در هر حالت تکه کاملی بدست آمده یا خیر
  - به همراه تکه

پویشگر جدول-گرا

مثال $r\ [0\ldots9]^+$

دارای چهاربخش

- آغازین
- حلقه مدل‌ساز رفتار خم
  - خواندن حرف بعدی و تقلید رفتار خم
  - توقف با ورود به حالت خطا
  - دو جدول CharCat و دلتا: جهت نگهداری اطلاعات خم
- حلقه برگشت هنگام گذرکردن از رشته‌های موردپذیرش خم
  - استفاده از پشته جهت برگرداندن پویشگر به آخرین حالت پذیرش
- بخش تفسیر و اعلام نتیجه

```
NextWord()
  state ← s0;
  lexeme ← " ";
  clear stack;
  push(bad);

  while (state≠se) do
    NextChar(char);
    lexeme ← lexeme + char;
    if state ∈ SA
        then clear stack;
    push(state);
    cat ← CharCat[char];
    state ← δ[state,cat];
  end;

  while(state ∉ SA and
        state≠bad) do
    state ← pop();
    truncate lexeme;
    RollBack();
  end;

  if state ∈ SA
    then return Type[state];
    else return invalid;
```

r	0, 1, 2, ..., 9	EOF	Other
Register	*Digit*	*Other*	*Other*

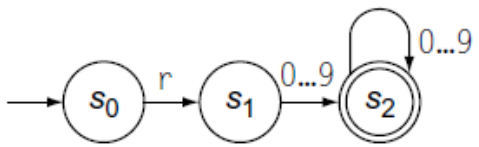
The Classifier Table, *CharCat*

	Register	*Digit*	*Other*
s_0	s_1	s_e	s_e
s_1	s_e	s_2	s_e
s_2	s_e	s_2	s_e
s_e	s_e	s_e	s_e

The Transition Table, δ

s_0	s_1	s_2	s_e
invalid	*invalid*	*register*	*invalid*

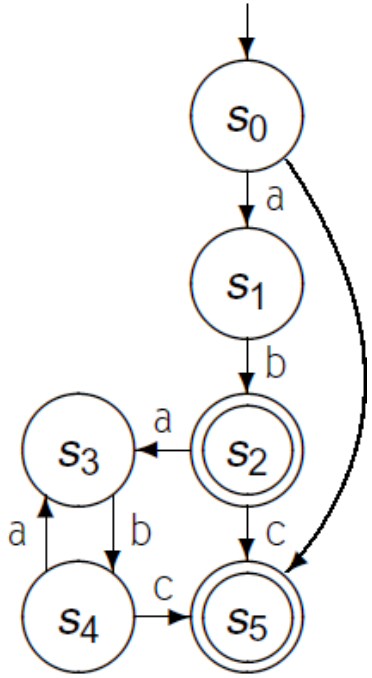
The Token Type Table, *Type*



The Underlying DFA

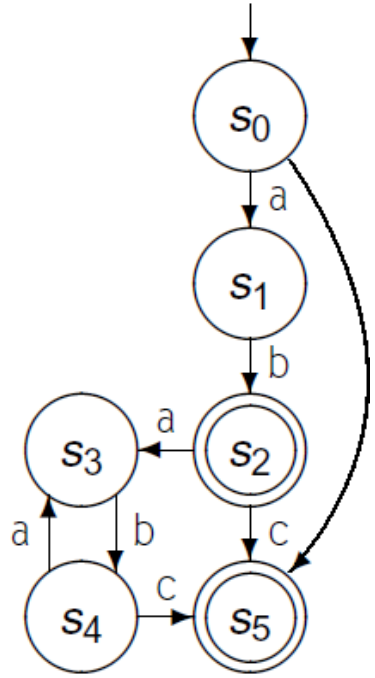
پویشگر جدول-گرا- پیش‌گیری از برگشت به عقب فراوان

مثال- $ab|(ab)^*c$ و رشته ورودی ababababc



پویشگر جدول-گرا- پیش‌گیری از برگشت به عقب فراوان

مثال- $ab|(ab)^*c$ و رشته ورودی ababababab

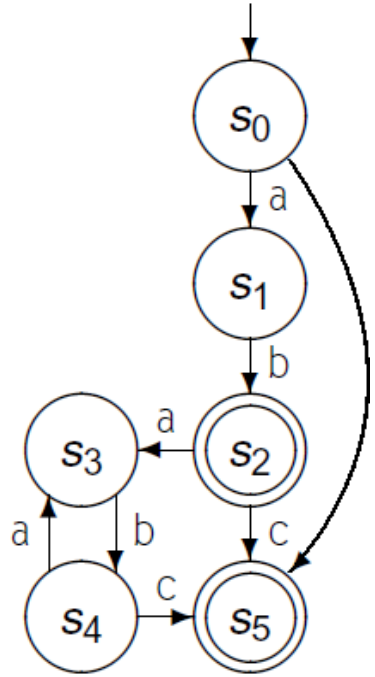


پویشگر جدول-گرا- پیش‌گیری از برگشت به عقب فراوان

```
NextWord()
  state ← s0 ;
  lexeme ← " ";
  clear stack;
  push(⟨bad, bad⟩);

  while (state≠se) do
    NextChar(char);
    InputPos ← InputPos + 1;
    lexeme ← lexeme + char;
    if Failed[state,InputPos]
        then break;
    if state ∈ SA
        then clear stack;
    push(⟨state,InputPos⟩);
    cat ← CharCat[char];
    state ← δ[state,cat];
  end;

  while(state ∉ SA and state≠bad ) do
    Failed[state,InputPos] ← true;
    ⟨state,InputPos⟩ ← pop();
    truncate lexeme;
    RollBack();
  end;

  if state ∈ SA
    then return TokenType[state];
    else return bad;
```

```
InitializeScanner()
    InputPos=0;
    for each state s in the DFA do
        for i=0 to |input stream| do
            Failed[s,i] ← false;
        end;
    end;
```

مثال- $ab\,|\,(ab)^*c$ و رشته و



منابع

[ببر سبز]

[اژدرها]

[کوپر]

[فیشر]

[نیسان]