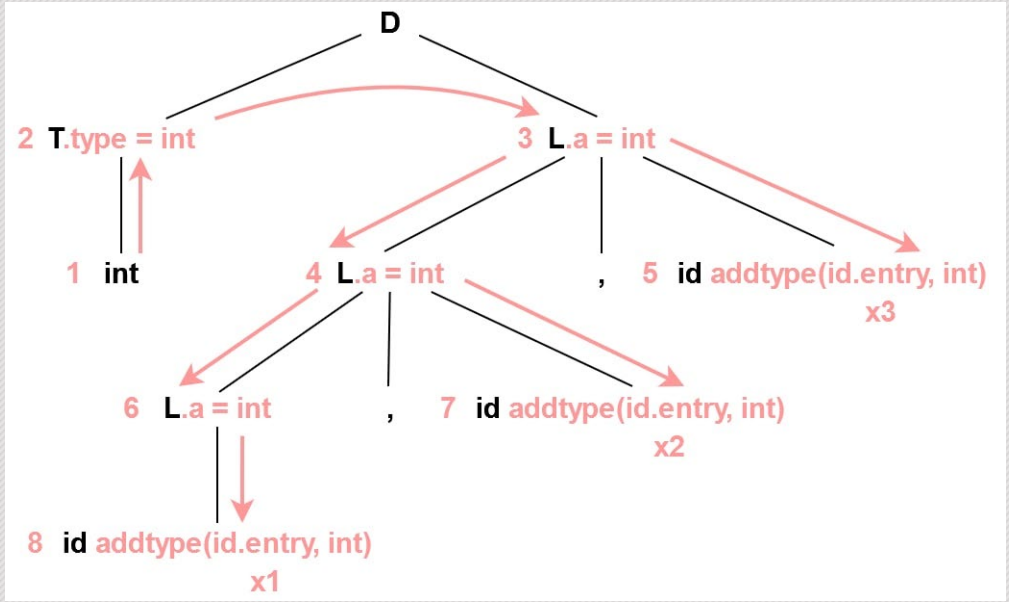
جواب تمرین سری پنجم درس کامپایلر

تهیه کنندگان:
زهرا اخلاقی
علیرضا صالحی حسین آبادی
استاد درس: زینب زالی

۱- الف


D
2 T.type = int
3 L.a = int
1 int
4 L.a = int
,
5 id addtype(id.entry, int)
x3
6 L.a = int
,
7 id addtype(id.entry, int)
x2
8 id addtype(id.entry, int)
x1

۱- الف(ادامه)

- اولین topological sort:
  - 1 2 3 4 6 8 7 5
- دومین topological sort:
  - 1 2 3 4 5 6 7 8

۱- ب

- در همان شکل الف مقادیر ویژگی‌ها نیز اضافه شده‌اند.

۱ـج

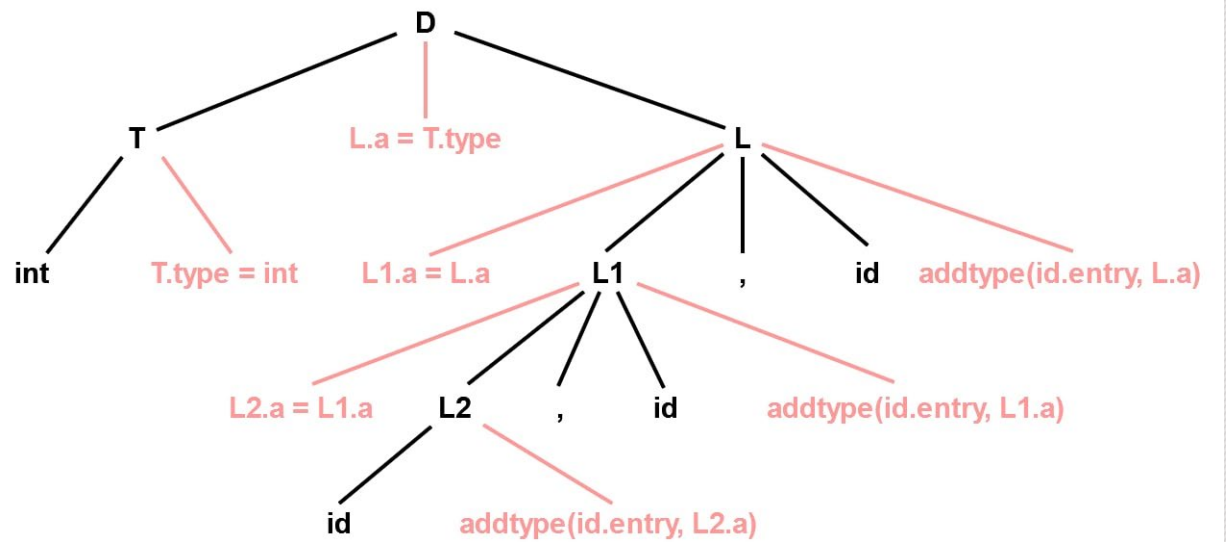
- تبدیل SDD به SDT(توضیح مراحل تبدیل صفحه ۸۸ جزوه کامپایلر ۳۹۸۲):

- D ⟶ T{L.a = T.type}L
- T ⟶ int{T.type = integer}
- T ⟶float{T.type = float}
- L ⟶ {L1.a = L.a} L1 , id{addType(id.entry, L.a)}
- L ⟶ id{addType(id.entry, L.a)}

۱–ج(ادامه)

- پیاده سازی به روش پیمایش درخت بعد از تجزیه:
  1. ایجاد درخت تجزیه بدون در نظر گرفتن اکشن‌ها
  2. به ازای هر گره میانی $A \rightarrow \alpha$، فرزندان دیگری به ازای همه اکشن‌ها درون α به این گره اضافه می‌کنیم به صورتی که فرزندان از چپ به راست به همان ترتیب اکشن‌ها و نمادها(سمبل‌ها) در Rule production مورد نظر باشند.
- درخت به صورت Preorder پیمایش می‌شود.

۱ –ج(ادامه)


D
T
L.a = T.type
L
int
T.type = int
L1.a = L.a
L1
,
id
addtype(id.entry, L.a)
L2.a = L1.a
L2
,
id
addtype(id.entry, L1.a)
id
addtype(id.entry, L2.a)

۲

- تبدیل SDD به SDT:

- $T \rightarrow F\{T'.inh = F.val\}\ T'\{T.val = T'.syn\}$
- $T' \rightarrow *F\{T'_1.inh = T'.inh \times F.val\}\ T'_1\{T'.syn = T'_1.syn\}$
- $T' \rightarrow \varepsilon\{T'.syn = T'.inh\}$
- $F \rightarrow digit\{F.val = digit.lexval\}$

۲ (ادامه)

- پیاده سازی Top-Down به روش Recursive descent:
  - وجود سه سمبل غیرنهایی در SDT = نوشتن سه تابع
  - T فقط دارای یک ویژگی ساختگی = تابع T فقط یک مقدار برگشتی
  - T′ دارای یک ویژگی موروثی و یک ویژگی ساختگی = تابع T′ دارای یک ورودی و یک مقدار برگشتی
  - F فقط دارای یک ویژگی ساختگی = تابع F فقط یک مقدار برگشتی

۲ (ادامه)

```
int T(){
        int T′inh, T′syn,Fval,Tval;
        Fval = F();
        T′inh = Fval;
        T′syn = T′(T′inh)
        Tval = T′syn;
        return Tval;
}
```

۲ (ادامه)

```
int T′ (int T′ inh){
        int T′1inh,T′1syn,Fval,Tval;
        If(lookahead == '*'){
                match('*');
                Fval = F();
                T′1inh = Fval;
                T′1syn = T′ (T′1inh);
                T′syn = T′1syn;
        }
        if(lookahead ∈ follow(T′)){
                T′syn = T′inh;
        }
        return T′syn ;
}
```

۲ (ادامه)

- طرح SDT معادل:

1. $T \rightarrow F\{M_1.inh = F.val\}M_1\{T'.inh = M_1.syn\}T\{T.val = T'.syn\}$
2. $T' \rightarrow *F\{M_2.inh = M_2.inh \times F.val\}M_2\{T_1'.inh = M_2.syn\}T_1'\{T'.syn = T_1'.syn\}$
3. $T' \rightarrow \varepsilon\{T'.syn = T'.inh(M1.val \text{ or } M2.val)\}$
4. $F \rightarrow digit\{F.val = digit.lexval\}$
5. $M_1 \rightarrow \varepsilon\{M_1.\ T'inh(val) = F.val\}$
6. $M_2 \rightarrow \varepsilon\{M_2.\ T'inh(val) = T'.inh * F.val\ \}$

- هنگام کاهش توسط 4، M1 یا M2 روی استک هستند.
- هنگام انجام کاهش‌های 5 و 6 نود F زیر نود M روی استک

۲ (ادامه)

- $T \rightarrow FM_1T$ {stack[top – 2].val = stack[top].syn; top = top – 2 }
- $M_1 \rightarrow \varepsilon$ {M_1node=new node();M_1node.syn=stack[top].val; stack.push(M_1node); top = top + 1}
- $T' \rightarrow *FM_2T_1'$ {stack[top – 3].syn=stack[top].syn;top = top - 3}
- $T' \rightarrow \varepsilon$ {T'node=new node(); T'node.syn=stack[top].val(M_1 or M_2); stack.push(T'node); top = top + 1}
- $M_2 \rightarrow \varepsilon$ {M_2node=new node();M_2node.syn=stack[top].val; stack.push(M_2node); top = top + 1
- F → digit{stack[top].val=symbotable.get(stack[top].entry).lexval}

۳

- $S \rightarrow L_1.L_2 \begin{cases} L_1.\text{isleft} = \text{true} \\ L_2.\text{isleft} = \text{false} \\ S.\text{val} = L_1.\text{val} + L_2.\text{val} \end{cases}$
- $S \rightarrow L \begin{cases} L.\text{isleft} = \text{true} \\ S.\text{val} = L.\text{val} \end{cases}$
- $L \rightarrow L_1 B \begin{cases} L_1.\text{isleft} = L.\text{isleft} \\ L.\text{len} = L_1.\text{len} + 1 \\ L.\text{val} = L.\text{isleft}? L_1.\text{val} \times 2 + B.\text{val} \\ : L_1 + B.\text{val} \times 2^{-L.\text{len}} \end{cases}$
- $L \rightarrow B \begin{cases} L.\text{len} = 1 \\ L.\text{val} = L.\text{isleft} \end{cases}$
- $B \rightarrow 0 \{B.\text{val} = 0$
- $B \rightarrow 1 \{B.\text{val} = 1$

مشخص شدن سمت چپ بودن یا نبودن Lها : ۱
اعشاری نبودن عدد و وجود نقطه در سمت چپ: ۲
محاسبه مقدار L با در نظر گرفتن مکان آن: ۳ و ۴
مشخص شدن ارزش بیت‌ها: ۵ و ۶
isleft: یک ویژگی موروثی، نشان دهنده وجود گره در سمت چپ نقطه اعشار(true: وجود دارد، false: وجود ندارد)
len: یک ویژگی ساختگی، نشان دهنده طول رشته باینری موجود در گره
val: یک ویژگی ساختگی، نگه‌داری مقدار گره

۴ – الف

$S \rightarrow \textbf{if} \; (\; C \;) \; S_1 \; \textbf{else} \; S_2$

- L1 = new()
- L2 = new()
- C.true = L1
- C.false = L2
- S1.next = S.next
- S2.next = S.next
- S.code = C.code || label ||L1 S1.code || label || L2 || S2.code

۴ – ب

$S \rightarrow \textbf{do } S_1 \textbf{ while } (C)$

- L1 = new()
- C.true = L1
- C.false = S.next
- S.code = label || L1 || S1.code || C.code

۴ - ج

$S \rightarrow '\{' \ L \ '\}'; L \rightarrow L \ S \mid \epsilon$

- L.next = new()
- S.code = L.code || label || L.next
- L1.next = new()
- S.next = L.next
- L.code = L1.code || label || L1.next || S.code

۵

M.instr: مقصد همه دستورات B.truelist
S.nextlist: شامل B.falselist و S1.nextlist

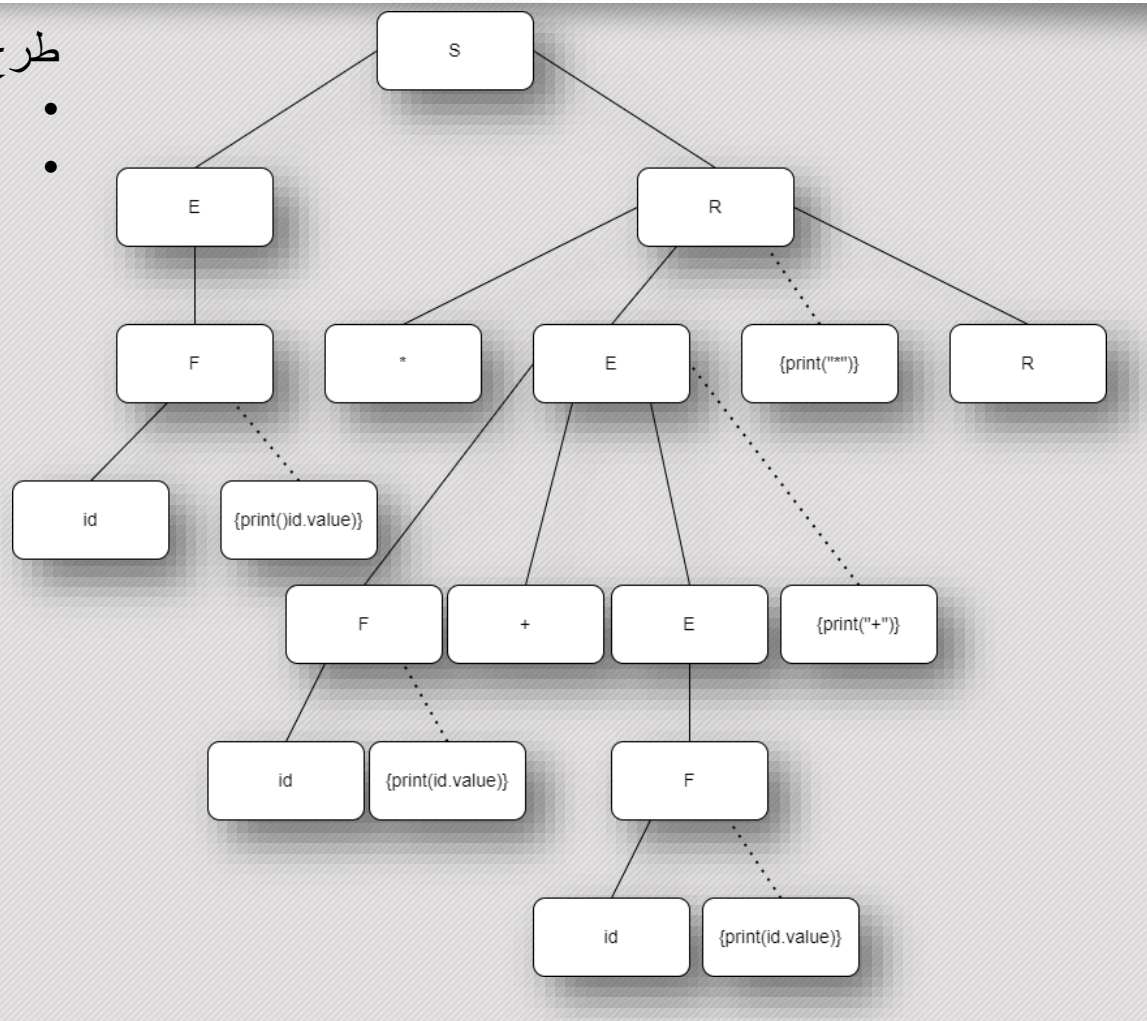
- S → If (B) M S1 { backpatch(B.truelist, M.instr);
  S.nextlist = S1.nextlist;
  backpatch(B.false,S.next) }
- B → B1 || M B2 { backpatch(B1.falselist, M.instr);
  B.truelist = merge(B1.truelist, B2.truelist);
  B.falselist = B2.falselist; }
- B → B1 && M B2 { backpatch(B1.truelist, M.instr);
  B.falselist = merge(B1.falselist, B2.falselist); }
- B → true { B.truelist = makelist(next.instr) ; gen('goto _'); }
- B → false { B.falselist = makelist(next.instr) ; gen('goto _'); }
- M → ϵ { M.instr = next.instr; }

۶

طرح ترجمه‌ای که چاپ می‌شود:

- عبارتی است postfix معادل عبارت infix
- از طریق dfs روی درخت تجزیه

234+*



۷

- تجزیه کننده LR حین تجزیه پایین به بالا در حالی که Actionها در انتهای Production rule هستند، actionهای مربوطه را اجرا می‌کنند. می‌توان با ساخت درخت تجزیه و پیمایش postorder درخت و محاسبه ویژگی‌های هنگام دیده شدن(ویزیت شدن) گره‌ها(نودها) ، عبارت ورودی را ترجمه کرد.

۷ (ادامه)

پیمایش postorder: ab+cd+.