Contents

Translation Grammars

- A context-free grammar in which the set of terminal symbols is partitioned into a set of input symbols and a setion of action symbols
- The strings in the language specified by a translation grammar are called activity sequences
- A context-free grammar may be converted into a translation grammar by inserting action symbols at appropriate locations within the productions from the context-free grammar
- A translation grammar in which all the action symbols specify output routines is termed a string translation grammar

Design a Translation Grammar to convert an arithmetic expression from infix to postfix form

- `A+B*C+D`
- `ABC*+D+`
- `A{A}+B{B}*C{C}{*}{+}+D{D}{+}`

- Represented the input and the output

- {A} - action symbol to output symbol A

Grammar

1. <E> $\rightarrow$ <E>+<T> {+}
2. <E> $\rightarrow$ <T>
3. <T> $\rightarrow$ <T>*<P> {*}
4. <T> $\rightarrow$ <P>
5. <P> $\rightarrow$ (<E>)
6. <P> $\rightarrow$ ident {ident}

```
A+B*C+D
```

- <E> $\overset{1}{\Rightarrow}$ <E>+<T> {+}
- <E>+<T> {+} $\overset{1}{\Rightarrow}$ <E>+<T>+<T> {+}{+}
- <E>+<T>+<T> {+}{+} $\overset{2}{\Rightarrow}$ <T>+<T>+<T> {+}{+}
- <T>+<T>+<T> {+}{+} $\overset{4}{\Rightarrow}$ <P>+<T>+<T> {+}{+}
- <P>+<T>+<T> {+}{+} $\overset{6}{\Rightarrow}$ $ident_A$+<T>+<T> {identA}{+}{+}
- $ident_A$+<T>+<T> {+}{+}{identA} $\overset{3}{\Rightarrow}$ $ident_A$+<T>*<P>+<T> {identA}{*}{+}{+}
- $ident_A$+<T>*<P>+<T> {+}{+}{identA}{*} $\overset{*}{\Rightarrow}$ $ident_A$+$ident_B$*$ident_C$+$ident_D$ {identA}{identB}{identC}{*}{+}{identD}{+}‘

E-list Grammar

1. <E> $\rightarrow$ <T><E-list>
2. <E-list> $\rightarrow$ +<T> {+}<E-list>
3. <E-list> $\rightarrow \varepsilon$
4. <T> $\rightarrow$ <P><T-list>
5. <T-list> $\rightarrow$ *<P> {*}<T-List>
6. <T-list> $\rightarrow \varepsilon$
7. <P> $\rightarrow$ (<E>)
8. <P> $\rightarrow$ ident {ident}

- A is an attribute

Synthesized Attributes

1. <S> $\rightarrow$ <E> {answer}
2. <E> $\rightarrow$ <E>+<T>

3. <E> $\rightarrow$ <T>
4. <T> $\rightarrow$ <T>*<P>
5. <T> $\rightarrow$ <P>
6. <P> $\rightarrow$ (<E>)
7. <P> $\rightarrow$ const

(1+2)*(3+4)

- <S> $\overset{1}{\Rightarrow}$ <E> {answer}
- <E> {answer} $\overset{3}{\Rightarrow}$ <T> {answer}
- <T> {answer} $\overset{4}{\Rightarrow}$ <T>*<P> {answer}
- <T>*<P> {answer} $\overset{5}{\Rightarrow}$ <P>*<P> {answer}
- <P>*<P> {answer} $\overset{6}{\Rightarrow}$ (<E>)*<P> {answer}
- (<E>)*<P> {answer} $\overset{2}{\Rightarrow}$ (<E>+<T>)*<P> {answer}
- (<E>+<T>)*<P> {answer} $\overset{3}{\Rightarrow}$ (<T>*<T>)+<P> {answer}
- (<T>*<T>)+<P> {answer} $\overset{5}{\Rightarrow}$ (<P>*<T>)+<P> {answer}
- (<P>*<T>)+<P> {answer} $\overset{7}{\Rightarrow}$ ($const_1$*<T>)+<P> {answer}
- ($const_1$*<T>)+<P> {answer} $\overset{*}{\Rightarrow}$ ($const_1$*$const_2$)+<P> {answer}
- ($const_1$*$const_2$)+<P> {answer} $\overset{*}{\Rightarrow}$ ($const_1$*$const_2$)+($const_3$*$const_4$) {answer}

  Attribute that goes up: synthesized

  Attributes that go accross: inherited

- $const_1$ synthesizes attribute <P>, which synthesizes attributes <T>, which synthesizes attribute <E>
- $const_2$ synthesizes attribute <P>, which synthesizes attributes <T>
- (<E>) is synthesized by <E> and <T>
- Eventually, {answer} inherits <E>
- {answer} - action symbol to print the value of inherited attribute

1+2*3

- <S> $\overset{1}{\Rightarrow}$ <E> {answer}
- <E> {answer} $\overset{2}{\Rightarrow}$ <E>+<T> {answer}
- <E>+<T> {answer} $\overset{4}{\Rightarrow}$ <E>+<T>*<P> {answer}
- <E>+<T>*<P> {answer} $\overset{3}{\Rightarrow}$ <T>+<T>*<P> {answer}
- <T>+<T>*<P> {answer} $\overset{5}{\Rightarrow}$ <P>+<T>*<P> {answer}
- <P>+<T>*<P> {answer} $\overset{7}{\Rightarrow}$ $const_1$+<T>*<P> {answer}

- $const_1$+<T>*<P> {answer} $\stackrel{5}{\Rightarrow}$ $const_1$+<P>*<P> {answer}
- $const_1$+<P>*<P> {answer} $\stackrel{7}{\Rightarrow}$ $const_1$+$const_2$*<P> {answer}
- $const_1$+$const_2$*<P> {answer} $\stackrel{7}{\Rightarrow}$ $const_1$+$const_2$*$const_3${answer}

Inherited Attributes

- Consider the Context Free Grammar:
  - <decl> $\rightarrow$ type ident <ident list>
  - <ident list> $\rightarrow$, ident <ident list>
  - <ident list> $\rightarrow \varepsilon$
- Where ident is a lexical token with
  - Class part “ident”
  - Value pointer to symbol table entry describing
    * The identifier
    * Type is a lexical token with class part “type” and value bool, bloat, int

1. <decl> $\rightarrow$ type ident {set type} <ident list>
2. <ident list> $\rightarrow$, ident {set type} <ident list>
3. <ident list> $\rightarrow \varepsilon$

- <decl> $\stackrel{1}{\Rightarrow}$ $type_{int}$ $ident_{pointerA}$ {set type} <ident list>

  {set type} inherits pointer A from ident and int from type

- $type_{int}$ $ident_{pointerA}$ {set type} <ident list> $\stackrel{2}{\Rightarrow}$ $type_{int}$ $ident_{pointerA}$ {set type} , $ident_{pointerB}$ {set type} <ident list>

  <ident list> inherits int

  {set type} inherits pointer B from ident and int from <ident list>

$A_1 * (B_2 + C_3)$

Symbol Table

	Name	Type	Address
1	A		
2	B		
3	C		
4			
5			

- Allocate a new symbol-table entry for (describing) a partial result
- *Infix* - A*(B+C)
- *Postfix* - ABC+*
- Activity Sequence
  - `A*(B+C {ADD}) {MULT}`
  - {$Add_{2,3,4}$} {$Mult_{1,4,5}$}

1. $< E > \rightarrow < E > + < T >$ {ADD}
2. $< E > \rightarrow < T >$
3. $< T > \rightarrow < T > * < P >$ {MULT}
4. $< T > \rightarrow < P >$
5. $< P > \rightarrow (< E >)$
6. $< P > \rightarrow ident$

- $< E > \overset{2}{\Rightarrow} < T >$
- $< T > \overset{3}{\Rightarrow} < T > * < P >$ {MULT}
- $< T > * < P >$ {MULT} $\overset{3}{\Rightarrow} < P > * < P >$ {MULT}
- $< P > * < P >$ {MULT} $\overset{6}{\Rightarrow} ident_1 * < P >$ {MULT}
- $ident_1 * < P >$ {MULT} $\overset{5}{\Rightarrow} ident_1 * (< E >)$ {MULT}
- $ident_1 * (< E >)$ {MULT} $\overset{1}{\Rightarrow} ident_1 * (< E > + < T >)$ {MULT} {ADD}
- $ident_1 * (< E > + < T >)$ {MULT} {ADD} $\overset{*}{\Rightarrow} ident_1 * (ident_2 + ident_3)$ {MULT} {ADD}

1. $< E >_p \rightarrow < E >_q + < T >_v$ {$ADD_{s,t,u}$}
   - $s \leftarrow q$
   - $t \leftarrow v$
   - $(p, u) \leftarrow NEWT$
2. $< E >_p \rightarrow < T >_q$

3. $< T >_p \rightarrow < T >_q * < P >_v$ $\{MULT_{s,t,u}\}$
    - $s \leftarrow q$
    - $t \leftarrow v$
    - $(p, u) \leftarrow NEWT$
4. $< T >_p \rightarrow < P >_q$
5. $< P >_p \rightarrow (< E >_q)$
6. $< P >_p \rightarrow ident_q$

   Where $< E >_p, < T >_p$ & $< P >_p$ synthesized p, all action symbol attributes are inherited and $NEWT$ allocated a new symbol table entry (for) describing a partial result

- Design Attributed Translations for:

1. $< E > \rightarrow < E >< ADDOP >< T >$ 2.
    - $< VARIABLE > \rightarrow ident$
    - $< VARIABLE > \rightarrow ident[< E >]$
2. $ident := < E >$

$< s > \rightarrow REPEAT < s > UNITL < c >$

- *REPEAT* and *UNTIL* aren't real - syntactic sugar

Flow

1. $EXECUTE < s >$
2. $EVALUATE < c >$
3. If False, back to 1

Translation

1. {LABEL}
2. $< s >$
3. $< c >$
4. $\{JUMPF_1\}$

- $< s > \rightarrow REPEAT$ $\{LABEL_p\}$ $< s > UNTIL < c >_q$ $\{JUMPF_{r,s}\}$
    - $r \leftarrow q$
    - $(p, s) \leftarrow NEWL$

   Where $< c >_p$ synthesized p, all action symbol attributes are inherited and $NEWL$ allocates a new symbol table entry for a table

$< s > \rightarrow IF < c > THEN < s >$

Flow

1. EVALUATE <c>
2. True? False?
3. If true, <s>
4. If false, skip 3

Translation

1. <c>
2. $\{\text{JUMPF}_4\}$
3. <s>
4. {LABEL}

- <s> → IF $\texttt{<c>}_{\text{p}}$ $\{\text{JUMPF}_{\text{q,r}}\}$ THEN <s> $\{\text{LABEL}_{\text{s}}\}$
  - q ← p
  - (r, s) ← NEW L

  Where $\texttt{<c>}_{\text{p}}$ synthesized p, all action symbol attributes are inherited and NEW L allocates a new symbol table entry for a table

Type	Label Value
LABEL	A00128...

Symbol Table

- for (r, s), both r and s pointing to the same entry in symbol table
- If your jump come before the label, it's not in the symbol table yet
- If only 0 in the symbol table, but the address of the following instruction in the symbol table
- When see the label, check if symbol table entry is 0
- If it is, label replaces placeholder value in object code

→ WHILE DO

Flow

1. EVALUATE <c>
2. True? False?
3. If true, <s>
4. Go back to 1
5. If false, skip 3 and 4

Translation

1. {LABEL}
2. <c>
3. $\{\text{JUMPF}_6\}$
4. <s>
5. $\{\text{JUMP}_1\}$
6. {LABEL}

- <s> $\rightarrow$ WHILE $\{\text{LABEL}_\text{t}\}$ <c>$_\text{p}$ $\{\text{JUMPF}_{\text{q,r}}\}$ DO <s> $\{\text{JUMP}_\text{u}\}$ $\{\text{LABEL}_\text{s}\}$
  - (t, u) $\leftarrow$ NEW L
  - q $\leftarrow$ p
  - (r, s) $\leftarrow$ NEW L

Sample Program

```
PROGRAM TEST(INPUT, OUTPUT)
VAR
    CH: CHAR.
    X, Y: REAL,
    I, J, K: INTEGER;
BEGIN

END.
```

- <program>$_\text{p}$ $\rightarrow$ PROGRAM IDENT_q (<IDENT LIST>); <DECLERATIONS> <COMPOUND STATEMENT>.
  - p $\leftarrow$ q
  - (<ident list>$_\text{p}$ $\{\text{SET FILE}\}_\text{q}$);...
    - p $\leftarrow$ q
  - <ident list>$_\text{p}$ $\rightarrow$ <ident list>$_\text{q}$, IDENT_r $\{\text{LINK ID}\}_{\text{s,t}}$
    - s $\leftarrow$ r
    - t $\leftarrow$ q

link id links the identifier pointed to by `s` *in front of the linked-list of identifiers pointed to by* `t`

- - <ident list>$_p \rightarrow$ IDENT$_q$
    - * $p \leftarrow q$
  - <declerations>$_p$ {ALLOCATE}$_q$
    - * $p \leftarrow q$
  - <declerations>$_p \rightarrow \varepsilon$
    - * $p \leftarrow$ NIL
  - <declerations>$_p \rightarrow$ VAR <dec list>$_q$
    - * <dec list>$_p \rightarrow$ <ident list>$_q$: <type>$_r$ {SET TYPE}$_{s,t}$ <more decs>$_u$

Name	TYPE	ADDRESS	…	ID LINK	DEC LINK
TEST					
INPUT	FILE				
OUTPUT	FILE				
CH	CHAR				
X	REAL				
Y	REAL				
I	INT				
J	INT				
K	INT				
…					

- ID Link starts as `null`

Consider the Context-Free Grammar

1. <S> $\rightarrow$ ident := <E>
2. <E> $\rightarrow$ ident <R>
3. <R> $\rightarrow$ + ident <R>
4. <R> $\rightarrow$ * ident <R>
5. <R> $\rightarrow \varepsilon$

```
X := A+B*C
```

- There's no precendence
- If using this to parse, you'll get
  - `A+B*C`
  - Not `A+(B*C)`

Translation Grammar

1. $\texttt{<S>} \rightarrow \text{ident} := \texttt{<E>}\ \{\text{assign}\}$
2. $\texttt{<E>} \rightarrow \text{ident}\ \texttt{<R>}$
3. $\texttt{<R>} \rightarrow +\ \text{ident}\ \{\text{add}\}\ \texttt{<R>}$
4. $\texttt{<R>} \rightarrow *\ \text{ident}\ \{\text{mul}\}\ \texttt{<R>}$
5. $\texttt{<R>} \rightarrow \varepsilon$

Attributed Translation Grammar

1. $\texttt{<S>} \rightarrow \text{ident}_{\text{p}} := \texttt{<E>}_{\text{q}}\ \{\text{assign}_{\text{r,s}}\}$
   - $\text{r} \leftarrow \text{q}$
   - $\text{s} \leftarrow \text{p}$
2. $\texttt{<E>}_{\text{p}} \rightarrow \text{ident}_{\text{q}}\ \texttt{<R>}_{\text{r,s}}$
   - $\text{r} \leftarrow \text{q}$
   - $\text{p} \leftarrow \text{s}$
3. $\texttt{<R>}_{\text{p,q}} \rightarrow +\ \text{ident}_{\text{r}}\ \{\text{add}_{\text{s,t,u}}\}\ \texttt{<R>}_{\text{v,w}}$
   - $\text{s} \leftarrow \text{p}$
   - $\text{t} \leftarrow \text{r}$
   - $(\text{v}, \text{u}) \leftarrow \text{NEW T}$
   - $\text{q} \leftarrow \text{w}$
4. $\texttt{<R>}_{\text{p,q}} \rightarrow *\ \text{ident}_{\text{r}}\ \{\text{mul}_{\text{s,t,u}}\}\ \texttt{<R>}_{\text{v,w}}$
   - $\text{s} \leftarrow \text{p}$
   - $\text{t} \leftarrow \text{r}$
   - $(\text{v}, \text{u}) \leftarrow \text{NEW T}$
   - $\text{q} \leftarrow \text{w}$
5. $\texttt{<R>}_{\text{p,q}} \rightarrow \varepsilon$
   - $\text{q} \leftarrow \text{p}$

   Where R_{p} synthesized p, all action symbol attributes are inherited, and NEW T allocated a new symbol table entry (for) describing a partial result

- Inherted attribute at ε begins to synthesize back up the tree
- Eventually `<E>` at top of tree is synthesized with attribute
- {assign} inherits the result

A*(B+C) - make derivation tree and attributions using e-list grammar