Traducción dirigida por sintaxis

Traducción descendente

Utilización de analizadores descendentes.

Utilización de esquemas de traducción.

Traducción dirigida por sintaxis

Traducción descendente

Eliminación de la recursividad por la izquierda.

```
L ::= E'\n'  {imprime(E.v)}
E ::= E1+T   {E.v = E1.v+T.v}
E ::= E1-T   {E.v = E1.v-T.v}
E ::= T      {E.v = T.v}
T ::= (E)    {T.v = E.v}
T ::= cte    {T.v = cte.valex}
```

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

$A ::= A_1 Y \quad \{A.a = g(A_1.a, Y.y)\}$

$A ::= X \quad \{A.a = f(X.x)\}$

Gramática

$A ::= XR$

$R ::= YR_1$

$R ::= \lambda$

EDT

$A ::= X\ R$

$R ::= Y\ R_1$

$R ::= \lambda$

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

$A ::= A_1 Y \quad \{A.a = g(A_1.a, Y.y)\}$

$A ::= X \quad \{A.a = f(X.x)\}$

Gramática

$A ::= XR$

$R ::= YR_1$

$R ::= \lambda$

EDT

$A ::= X\{R.h=f(X.x)\}R$

$R ::= Y\ R_1$

$R ::= \lambda$

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

$A ::= A_1 Y \quad \{A.a = g(A_1.a, Y.y)\}$

$A ::= X \quad \{A.a = f(X.x)\}$

Gramática	EDT
$A ::= XR$	$A ::= X\{R.h=f(X.x)\}R$
$R ::= YR_1$	$R ::= Y\{R1.h=g(R.h,Y.y)\}R_1$
$R ::= \lambda$	$R ::= \lambda$

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

$A ::= A_1 Y \quad \{A.a = g(A_1.a, Y.y)\}$

$A ::= X \quad \{A.a = f(X.x)\}$

Gramática	EDT
$A ::= XR$	$A ::= X\{R.h=f(X.x)\}R$
$R ::= YR_1$	$R ::= Y\{R_1.h=g(R.h,Y.y)\}$ R_1
$R ::= \lambda$	$R ::= \lambda\{R.s=R.h\}$

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

```
A ::= A1Y     {A.a = g(A1.a, Y.y)}
A ::= X       {A.a = f(X.x)}
```

Gramática	EDT
A ::= XR	A ::= X{R.h=f(X.x)}R
R ::= YR$_1$	R ::= Y{R$_1$.h=g(R.h,Y.y)} R$_1$**{R.s= R$_1$.s}**
R ::= λ	R ::= λ{R.s=R.h}

Traducción dirigida por sintaxis

Traducción descendente

Generalización de la eliminación de la recursividad por la izquierda.

```
A ::= A1Y     {A.a = g(A1.a, Y.y)}
A ::= X       {A.a = f(X.x)}
```

Gramática	EDT
A ::= XR	A ::= X{R.h=f(X.x)}R**{A.a=R.s}**
R ::= YR_1	R ::= Y{R_1.h=g(R.h,Y.y)} R_1{R.s= R_1.s}
R ::= λ	R ::= λ{R.s=R.h}

Traducción dirigida por sintaxis

Traducción descendente

Ejercicio: Eliminar de la recursividad a izquierdas.

```
L ::= E'\n'  {imprime(E.v)}
E ::= E1+T   {E.v = E1.v+T.v}
E ::= E1-T   {E.v = E1.v-T.v}
E ::= T      {E.v = T.v}
T ::= (E)    {T.v = E.v}
T ::= cte    {T.v = cte.valex}
```

Traducción dirigida por sintaxis

Traducción descendente

Construcción de traductores recursivos

Similar los analizadores descendentes recursivos:

Sintetizados de NT =	Valor devuelto función NT.
Heredados de NT =	Parámetro de la función NT.
At.Terminales =	Variable para su valor.
Acciones semánticas =	Sentencias de código.

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= aBb
B ::= aBb | c
```

```
void S(void)
{
   cmp(a);
   B();
   cmp(b);
}
```

```
void B(void)
{
   if(entrada == a){
      cmp(a);
      B();
      cmp(b);
   } else
      if(entrada == c)
         cmp(c);
      else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

$$S ::= \{B.h=1;\}aBb\{imprime(B.s);\}$$

$$B ::= a\{B_1.h=B.h+1;\}B_1\{B.s=B_1.s\}b \mid c\{B.s=B.h*c.lex;\}$$

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}

        void S(void)
        {
          cmp(a);B();cmp(b);
        }
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
```

```
void S(void)
{
  int B_h;
  B_h=1;
  cmp(a);B();cmp(b);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void S(void)
{
  int B_h;
  B_h=1;
  cmp(a);B();cmp(b);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void S(void)
{
  int B_h;
  B_h=1;
  cmp(a);B(B_h);cmp(b);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void B(int B_h)
{
  if(entrada == a){
    cmp(a);B();cmp(b);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}

    void B(int B_h)
    {
      if(entrada == a){
        cmp(a);B();cmp(b);
      } else if(entrada == c) cmp(c);
             else error();
    }
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}

    void B(int B_h)
    {
      int B1_h;
      if(entrada == a){
        cmp(a);B(B1_h);cmp(b);
      } else if(entrada == c) cmp(c);
             else error();
    }
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void B(int B_h)
{
  int B1_h;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B(B1_h);cmp(b);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}

    void B(int B_h)
    {
      int B1_h;
      if(entrada == a){
        cmp(a); B1_h=B_h+1;
        B(B1_h);cmp(b);
      } else if(entrada == c) cmp(c);
             else error();
    }
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
int B(int B_h)
{
  int B1_h;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B(B1_h);cmp(b);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
int B(int B_h)
{
  int B1_h, B1_s;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B1_s=B(B1_h);cmp(b);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
int B(int B_h)
{
  int B1_h, B1_s;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B1_s=B(B1_h);cmp(b);return(B1_s);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
int B(int B_h)
{
  int B1_h, B1_s;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B1_s=B(B1_h);cmp(b);return(B1_s);
  } else if(entrada == c) cmp(c);
         else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
int B(int B_h)
{
  int B1_h, B1_s, c_lex;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B1_s=B(B1_h);cmp(b);return(B1_s);
  } else if(entrada == c){
              cmp(c);return(B_h*c_lex);
          } else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void S(void)
{
  int B_h;
  B_h=1;
  cmp(a);B(B_h);cmp(b);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void S(void)
{
  int B_h, B_s;
  B_h=1;
  cmp(a);B_s=B(B_h);cmp(b);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
```

```
void S(void)
{
  int B_h, B_s;
  B_h=1;
  cmp(a);B_s=B(B_h);cmp(b);
  imprime(B_s);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
S ::= {B.h=1;}aBb{imprime(B.s);}
```

```
void S(void)
{
  int B_h, B_s;
  B_h=1;
  cmp(a);
  B_s=B(B_h);
  cmp(b);
  imprime(B.s);
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
B ::= a{B1.h=B.h+1;}B1{B.s=B1.s}b |
      c{B.s=B.h*c.lex;}
int B(int B_h)
{
  int B1_h, B1_s, c_lex;
  if(entrada == a){
    cmp(a); B1_h=B_h+1;
    B1_s=B(B1_h);cmp(b);return(B1_s);
  } else if(entrada == c){
              cmp(c);return(B_h*c_lex);
         } else error();
}
```

Traducción dirigida por sintaxis

Traducción descendente

```
L ::= E'\n' {imprime(E.v);}
E ::= T {R.h=T.v;} R {E.v=R.v;}
R ::= +T {R1.h=R.h+T.v;} R1 {R.v=R1.v;}
R ::= -T {R1.h=R.h-T.v;} R1 {R.v=R1.v;}
R ::= λ {R.v=R.h;}
T ::= (E) {T.v=E.v;}
T ::= cte {T.v=cte.valex;}
```

Traducción dirigida por sintaxis

Traducción ascendente

Utilización de analizadores ascendentes

Funcionamiento por reducciones

Problemas:

- Todas LL(1) y no todas LR(1)
- Tratamiento de **atributos heredados**
- **Acciones intercaladas** en consecuentes

Traducción dirigida por sintaxis

Traducción ascendente

Acciones intercaladas en el consecuente

- Pasar acciones **al final** de un consecuente
- Convertir toda producción de la forma:

  `A ::= α{accion();}β`

- En otras dos de la forma:

  `A ::= αXβ`

  `X ::= λ {accion();}`

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Las definiciones utilizadas son:

  "con atributos **por la izquierda**"

- Se utiliza la idea de la **pila de atributos**
- El símbolo del que se hereda el atributo se encuentra en la pila

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Ejemplo: *bool id, id*

```
D ::= T {L.h=T.t;} L
T ::= real {T.t=r;}
T ::= bool {T.t=b;}
L ::= id {añadetipo(id.ent, L.h);}
L ::= {L1.h=L.h;} id ,
      {añadetipo(id.ent, L.h);} L1
```

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Problema: Localizar el símbolo del que se hereda
- Ejemplo:

```
S ::= aAC        {C.h=A.s;}
S ::= bABC       {C.h=A.s;}
C ::= c          {C.s=f(C.h);}
```

- Conseguir que el **A** y **C** estén siempre a la misma distancia...

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Problema: Localizar el símbolo del que se hereda
- Ejemplo:

```
S ::= aAC        {C.h=A.s;}
S ::= bABC       {C.h=A.s;}
C ::= c          {C.s=f(C.h);}
```

- Conseguir que el **valor de `A.s`** y **`C`** estén siempre a la misma distancia

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Conversión de atributos heredados en sintetizados

- Ejemplo:

```
D ::= L:T
T ::= integer | char
L ::= L, id | id
```

Traducción dirigida por sintaxis

Traducción ascendente

Tratamiento de atributos heredados

- Conversión de atributos heredados en sintetizados
- Cambio en la gramática
- Ejemplo:

```
D ::= id, D | id:T
T ::= integer | char
```