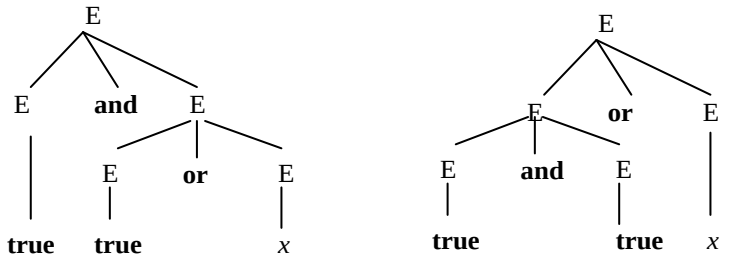
Algunos patrones para la especificación sintáctica

Especificación de expresiones

- Consideraremos expresiones formadas a partir de operadores unarios (prefijos y posfijos), así como de operadores binarios infijos (el tratamiento de otro tipo de operadores puede llevarse a cabo utilizando un análisis similar)
- La formalización *directa* conduce a gramáticas ambiguas. Por ejemplo: expresiones booleanas constituidas por literales **true** y **false**, variables (identificadores), los operadores binarios **and** y **or**, y el operador unario **not**:
  E → **true** | **false** | iden | E **and** E | E **or** E | **not** E
  Por ejemplo, la sentencia `true and true or x` tiene dos árboles sintácticos posibles:



- La forma de resolver esta ambigüedad es
  - Asociar *prioridades* y *asociatividades* a los operadores:
    - Cada operador tiene un nivel de prioridad (suponemos que los niveles son consecutivos y comienzan en 0). En ausencia de paréntesis, los operadores más prioritarios se aplican *antes* que los menos prioritarios.
    - Los operadores unarios pueden ser *asociativos* (es posible encadenar varios operadores del mismo nivel: `+ + + + 5`), o *no asociativos* (no es posible realizar dicho encadenado sin utilizar paréntesis: `+(+(+(+ 5)))`)
    - Los operadores binarios (infijos) pueden: asociar a izquierdas (`a+a+a` ≡ `(a+a)+a`), asociar a derechas (`a+a+a` ≡ `a+(a+a)`), o no asociar (`a+a+a` es sintácticamente errónea)
  - Introducir un no terminal distinto para cada nivel de prioridad (*E*0, *E*1, … *E*N)
  - Las reglas de la gramática se definen como sigue:
    - En el nivel de prioridad *i* Los operadores unarios prefijos asociativos θ dan lugar a producciones de la forma $E_i \rightarrow \theta E_i$. Los unarios prefijos no asociativos θ dan lugar $E_i \rightarrow \theta E_{i+1}$
    - Los operadores unarios postfijos asociativos θ dan lugar a producciones de la forma $E_i \rightarrow E_i\theta$, y los no asociativos a $E_i \rightarrow E_{i+1}\theta$
    - Los operadores binarios (infijos) θ asociativos a izquierdas dan lugar a producciones de la forma $E_i \rightarrow E_i\ \theta E_{i+1}$. Los que asocian a derechas a $E_i \rightarrow E_{i+1}\ \theta E_i$. Y los no asociativos a $E_i \rightarrow E_{i+1}\ \theta E_{i+1}$
    - En cada nivel se incluye una regla $E_i \rightarrow E_{i+1}$

        - Si *n* es el último nivel de prioridad, se incluye un no terminal E_{n+1} al que se asocian reglas para definir las expresiones básicas (números, identificadores, …), y también una regla de la forma $E_{n+1} \rightarrow (E_0)$
- En la gramática resultante, los árboles sintácticos reflejan estructuralmente las prioridades y asociatividades de los operadores
- Para evitar la ambigüedad, un operador con la misma aridad no puede aparecer en más de un nivel de prioridad. Así mismo, en cada nivel de prioridad:
    - Si existen operadores unarios prefijos asociativos, no pueden existir operadores unarios posfijos asociativos, ni operadores binarios (infijos) que asocien a izquierdas.
    - La existencia de operadores unarios posfijos asociativos prohíbe también la existencia de operadores binarios (infijos) asociativos a derechas.
    - La existencia de operadores binarios que asocien a izquierdas prohíbe también la existencia de operadores binarios que asocien a derechas.
- Ejemplo: Diseñar una gramática para un lenguaje de expresiones cuyas expresiones básicas son números, identificadores, y que involucra operadores con las siguientes características:

	Tipo	Prioridad	Asociatividad
*	Unario prefijo	0	Asociativo
&	Unario postfijo	0	No asociativo
-	Unario prefijo	0	No asociativo
&	Binario infijo	1	Asociativo a derechas
%	Binario infijo	1	No asociativo
!	Binario infijo	2	Asociativo a izquierdas

Solución:

E0 → * E0 | E1 & | - E1 | E1

E1 → E2 & E1 | E2 % E2 | E2

E2 → E2 ! E3 | E3

E3 → num | iden | (E0)

Especificación de secuencias

- Utilizar recursión a izquierdas (si la implementación final es ascendente, es más eficiente; si es descendente, hay un acondicionamiento directo que produce una gramática recursiva a derechas equivalente)
- Secuencia de elementos con un separador ϕ: equivalente a expresar un lenguaje de expresiones con ϕ como un operador infijo asociativo a izquierdas. Ejemplo: secuencia de instrucciones separadas por ;:

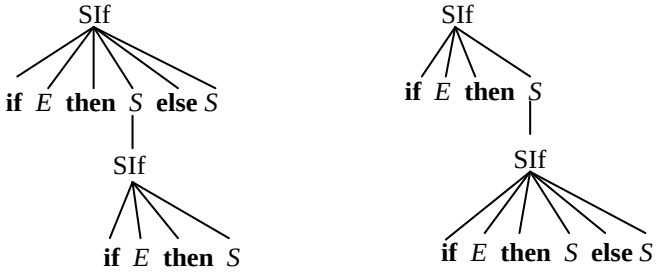
  Is → Is ; I | I
- Secuencia de elementos con un terminador ϕ: equivalente a expresar un lenguaje de expresiones con un operador infijo asociativo a izquierdas vacío (el terminador es parte de los elementos). Ejemplo: secuencia de instrucciones terminadas en ;

  Is → Is C | C

  C → I ;

Tratamiento de la ambigüedad *if – then - else*

- Supóngase la sintaxis del *if – then – else* definida como:
  SIf → **if** *E* **then** *S* | **if** *E* **then** *S* **else** *S*
  donde *S* representa las diferentes sentencias del lenguaje (entre ellas, *SIf*):
  S → *SAsig* | *SWhile* | … | *SIf*
  La gramática resultante es ambigua:
  if *x=5* **then if** *y=6* **then** *x:=x+1* **else** *x:=x-1*

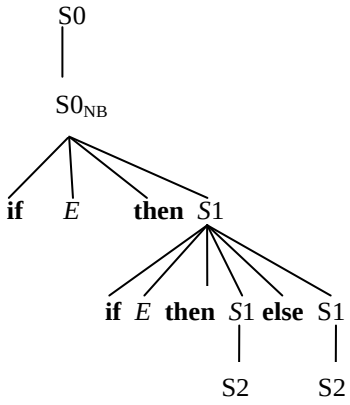


- La forma de evitar la ambigüedad es, de nuevo, por estratificación.
  - Si se desea, como es usual, que el *else* siempre esté asociado a los *if* más internos:

  S0 → $S0_{NB}$ | S1
  S1 → **if** *E* **then** *S1* **else** *S1* | S2
  S2 → *SAsig* | *SWhile* | … //todas las sentencias menos el *if*

  $S0_{NB}$ → **if** *E* **then** $S0_{NB}$ | **if** *E* **then** S1 | **if** *E* **then** *S1* **else** $S0_{NB}$

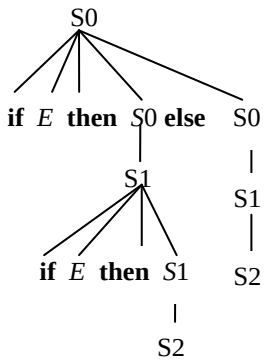
  ($S0_{NB}$, *sentencia de nivel 0 no balanceada*, representa las sentencias que contienen algún *if* sin parte *else*
  Puede comprobarse que la gramática es LR(1) y LALR(1). Puede probarse también a transformar la gramática para obtener una gramática LL(1) equivalente)



  - Si se desea, por el contrario, que el *else* siempre esté asociado a los *if* más externos:

S0 → **if** *E* **then** S0 **else** S0 | S1
S1 → **if** *E* **then** S1 | S2

S2 → *SAsig* | *SWhile* | … //todas las sentencias menos el *if*



(en este caso, la gramática que resulta no es de utilidad práctica. Por ejemplo, no es LR(k) para ningún k -*¿cómo saber, con un anticipo de longitud prefijada, que se han acabo ya los ifs con partes else?*)

- En la práctica, para evitar complicar la gramática tratando el *if* de manera especial se admite la ambigüedad, y se resuelven los conflictos a nivel de implementación.
- Otra opción es explicitar claramente el final del *if*:

SIf → **if** *E* **then** *S* **endif** | **if** *E* **then** *S* **else** *S* **endif**

Ejercicios

1) Formaliza con una gramática incontextual la sintaxis de lenguaje de expresiones con las características indicadas. Considera como expresiones básicas los números e identificadores. Contempla el uso de paréntesis para variar la prioridad y asociatividad de los operadores.

a)

	Tipo	Prioridad	Asociatividad
a	Binario infijo	3	No asociativo
b	Unario postfijo	3	Asociativo
c	Unario Infijo	2	Asociativo a derechas
d	Binario infijo	1	Asociativo a izquierdas
e	Unario prefijo	0	No asociativo

b)

	Tipo	Prioridad	Asociatividad
=	Binario infijo	0	No asociativo
*	Unario prefijo	2	Asociativo
++	Binario infijo	2	Asociativo a izquierdas
&&	Binario infijo	1	Asociativo a derechas

(Demuestra, además, que en este caso existe ambigüedad)

2) La siguiente gramática caracteriza la sintaxis de un lenguaje de expresiones:

```
A ::= B @ B
A ::= B + A
A ::= B
B ::= B <> C
B ::= + C
B ::= C
C ::= C +
C ::= D
D ::= n
D ::= (A)
```

Sabiendo que dicha gramática se ha construido utilizando los patrones habituales de representación de prioridades y asociatividades de operadores, se pide determinar el tipo (prefijo, infijo, postfijo), la prioridad (indicando claramente cuándo un operador tiene mayor, menor o la misma prioridad que otro), y la asociatividad de cada operador.