Vamos a descomponer y explicar cada una de estas reglas:

**Regla 1: E -> T E'**

Esta regla dice que una expresión (E) se puede descomponer en un término (T) seguido de E'.

* **T**: Un término.
* **E'**: Puede ser otra parte de la expresión que se maneja en la siguiente regla.

**Regla 2: E' -> + T E' | ε**

Esta regla define E' (la continuación de una expresión):

* + T E': Esta parte de la regla indica que E' puede ser un operador de suma (+) seguido por otro término (T) y luego otra posible continuación de la expresión (E').
* ε: Esta es la cadena vacía (epsilon), que indica que E' puede no tener nada más (termina aquí).

**Regla 3: T -> F T'**

Esta regla dice que un término (T) se descompone en un factor (F) seguido de T'.

* **F**: Un factor.
* **T'**: Puede ser otra parte del término que se maneja en la siguiente regla.

**Regla 4: T' -> \* F T' | ε**

Esta regla define T' (la continuación de un término):

* \* F T': Esta parte de la regla indica que T' puede ser un operador de multiplicación (\*) seguido por otro factor (F) y luego otra posible continuación del término (T').
* ε: Esta es la cadena vacía (epsilon), que indica que T' puede no tener nada más (termina aquí).

**Regla 5: F -> ( E ) | num**

Esta regla define un factor (F):

* ( E ): Un factor puede ser una expresión completa (E) encerrada entre paréntesis.
* num: Un factor también puede ser un número.

**Ejemplo de Parseo**

Veamos cómo se analiza una expresión concreta, como 3 + 5 \* (2 + 4) utilizando esta gramática.

1. **Expresión Inicial:** 3 + 5 \* (2 + 4)
   * Comenzamos con E.
2. **E -> T E':** Necesitamos un término seguido de E'.
   * T empieza con 3.
3. **T -> F T':** Necesitamos un factor seguido de T'.
   * F es 3.
4. **T' -> ε:** No hay más multiplicaciones en 3, así que terminamos el término aquí.
5. **E' -> + T E':** Continuamos con + seguido de otro término (T).
6. **T -> F T':** Después del +, el siguiente término empieza con 5.
7. **F -> num:** 5 es un número, así que es un factor.
8. **T' -> \* F T':** 5 es seguido por \* y otro factor.
9. **F -> ( E ):** Después del \*, tenemos una expresión entre paréntesis (2 + 4).
10. **E -> T E':** Dentro de los paréntesis, empezamos con una expresión E que es 2 + 4.
11. **T -> F T':** El término T es 2.
12. **F -> num:** 2 es un número, así que es un factor.
13. **T' -> ε:** No hay más multiplicaciones en 2, así que terminamos el término aquí.
14. **E' -> + T E':** Después del 2, tenemos un + y otro término 4.
15. **T -> F T':** El término T es 4.
16. **F -> num:** 4 es un número, así que es un factor.
17. **T' -> ε:** No hay más multiplicaciones en 4, así que terminamos el término aquí.
18. **E' -> ε:** No hay más sumas en 2 + 4, así que terminamos la expresión aquí.

Así, hemos descompuesto completamente la expresión 3 + 5 \* (2 + 4) siguiendo las reglas de la gramática.

**Visualización del Árbol de Sintaxis Abstracta (AST)**

El árbol de sintaxis abstracta para 3 + 5 \* (2 + 4) sería algo así:

markdown

Copiar código

+

/ \

3 \*

/ \

5 +

/ \

2 4

Cada nodo del árbol representa una operación y sus hijos son los operandos de esa operación. Este árbol se genera siguiendo las reglas de la gramática de manera recursiva.