Ejercicio 4. Para un lenguaje libre de contexto de libre elección, comprobar que existe ambigüedad en su gramática encontrando dos árboles de derivación para una misma cadena del lenguaje.

El primer lenguaje seleccionado sería: **L = a – b – c**. Este lenguaje tiene la peculiaridad de ser ambiguo, pero de tener una solución que no lo es; es decir, existen gramáticas que lo resuelven que no son ambiguas, como también ocurre con lenguajes como **L = { a^{2+3i} : i >= 0}**, cuya solución ambigua es:

$$S \longrightarrow AA, \quad S \longrightarrow aSa, \quad A \longrightarrow a$$

Y su solución no ambigua es:

$$S \longrightarrow aa, \qquad S \longrightarrow aaU, \qquad U \longrightarrow aaaU, \qquad U \longrightarrow aaa$$

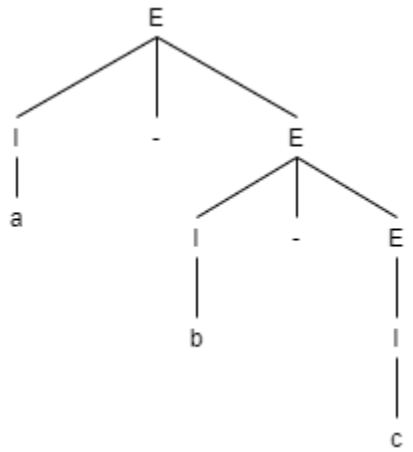
Ahora pasamos al estudio de la ambigüedad del primer problema y al procedimiento para hacerlo no ambiguo. La gramática que genera cadenas del tipo ***a – b – c*** es la siguiente:

$$E \longrightarrow I, \qquad E \longrightarrow I - E, \qquad E \longrightarrow E - I, \qquad I \longrightarrow a \mid b \mid c \mid d$$

Tendiendo como posibles soluciones válidas tanto:

$$E \longrightarrow I - E \longrightarrow a - I - E \longrightarrow a - b - I \longrightarrow a - b - c$$

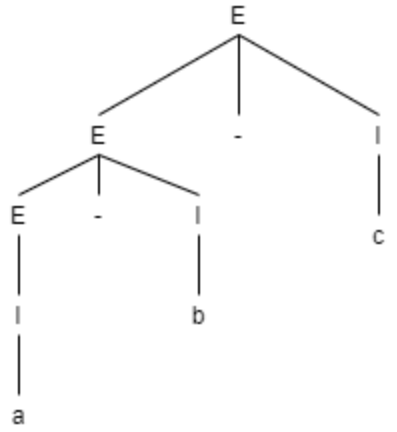
Con su árbol correspondiente:



Como:

$$E \longrightarrow E - I \longrightarrow E - I - c \longrightarrow I - b - c \longrightarrow a - b - c$$

Con su árbol correspondiente:



Al tener dos árboles de derivación diferentes, cada uno con su propia estructura sintáctica, podemos decir que el lenguaje es ambiguo. La incertidumbre real reside en no poder concretar si, como en la primera imagen, se resta a "a" el conjunto "b-c", siendo a – (b - c); o, por el contrario, como en la segunda imagen, se resta a "a-b" la unidad "c", quedando como (a - b) – c. El resultado terminaría siendo el mismo en un caso que en otro, pero la sintaxis no es la misma en ambas situaciones.

Una forma de solucionar la ambigüedad sería eliminar de todas las expresiones la que es:

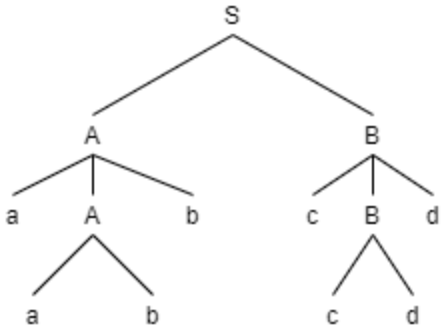
E ⟶ I – E , lo cual haría desaparecer el primer árbol y solo se podría formar el segundo. Esto despeja la duda, la solución real sería: **(a - b) – c**.

Ahora que sabemos que la gramática de un lenguaje libre del contexto puede ser ambigua, pasemos al caso en el que generemos la gramática que generemos, lo seguirá siendo inherentemente.

Es el caso del lenguaje $L = \{a^n b^n c^m d^m : n >= 1, m >= 1\} U \{ a^n b^m c^m d^n : n >= 1, m >= 1\}$, cuya gramática es:

$$S \longrightarrow AB, \quad A \longrightarrow ab, \quad A \longrightarrow aAb, \quad B \longrightarrow cd,$$
$$B \longrightarrow cBd, \quad S \longrightarrow aCd, \quad C \longrightarrow aCd, \quad C \longrightarrow bDc,$$
$$C \longrightarrow bc, \quad D \longrightarrow bDc, \quad D \longrightarrow bc$$

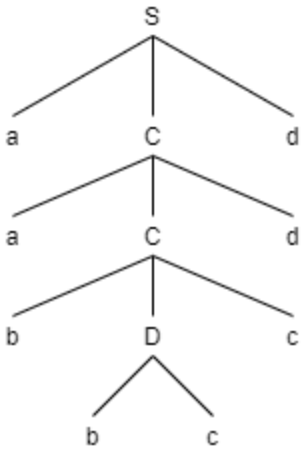
Por lo que, para la cadena de ejemplo *[aabbccdd]*, existen estos dos árboles de derivación distintos:



Asociado a la solución:

$$S \longrightarrow AB \longrightarrow aAbcBd \longrightarrow aabbccdd$$

Y este:



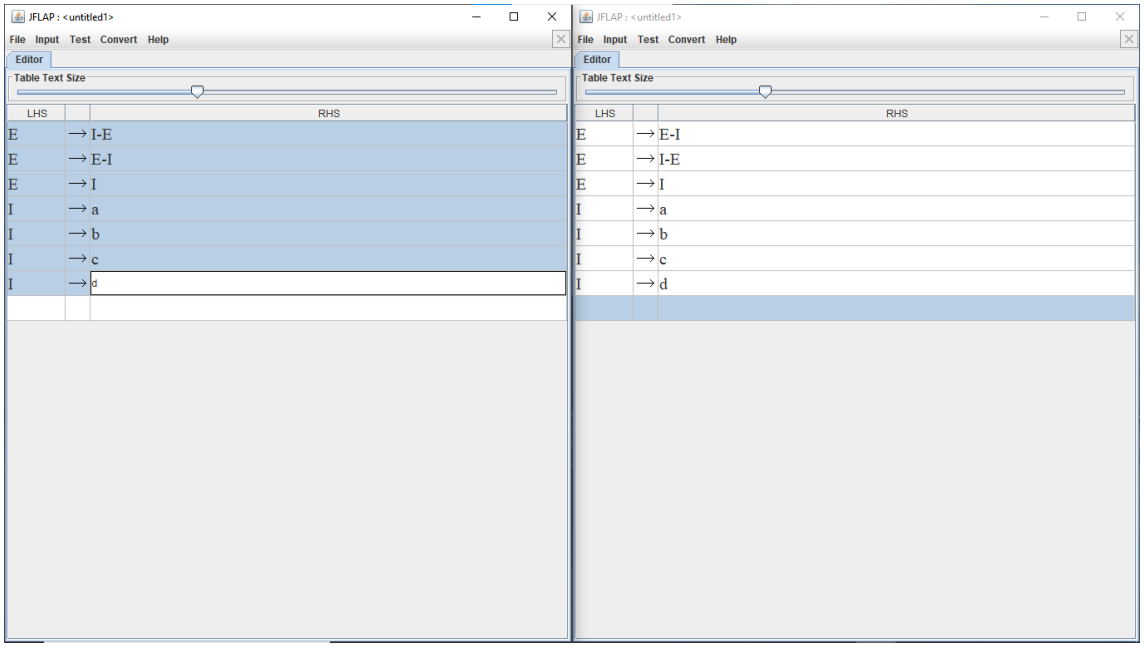
Asociado a la solución:

$$S \longrightarrow aCd \longrightarrow aaCdd \longrightarrow aabDcdd \longrightarrow aabbccdd$$

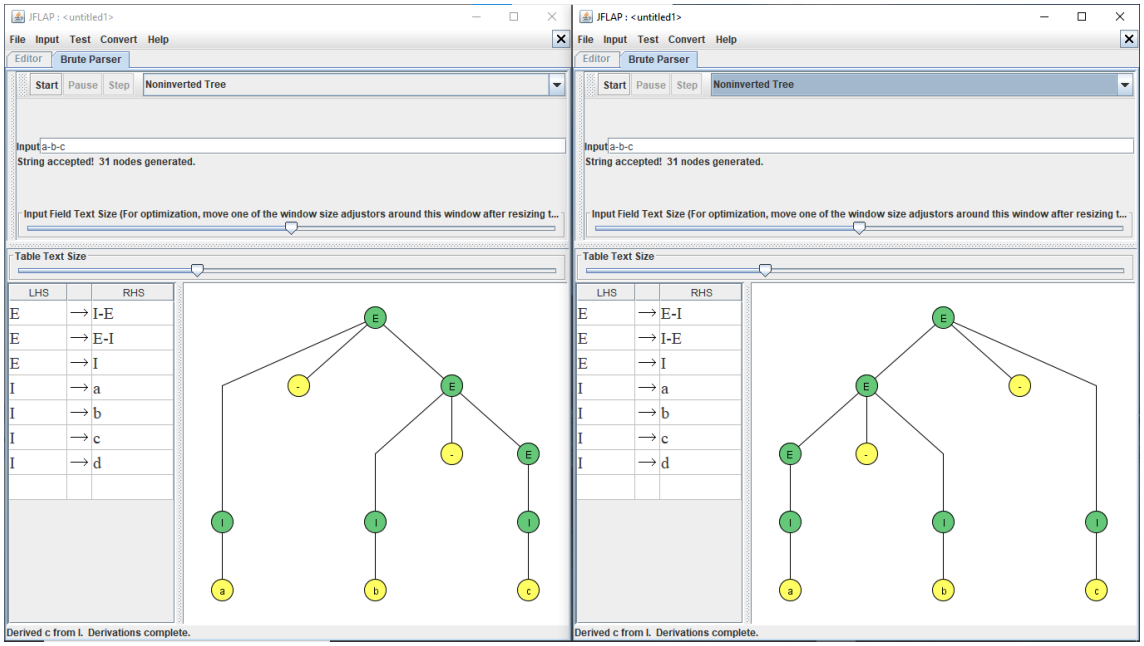
Así, demostramos que es inherentemente ambiguo, ya que no existe modificación en la gramática que genere una solución con un solo árbol de derivación.

Ahora vamos a utilizar **JFlap** para demostrar que la modificación en el primer lenguaje escogido ($L = a - b - c$) arregla la ambigüedad y que el segundo lenguaje escogido ($L = \{a^n b^n c^m d^m : n >= 1, m >= 1\} U \{a^n b^m c^m d^n : n >= 1, m >= 1\}$) es inherentemente ambiguo.

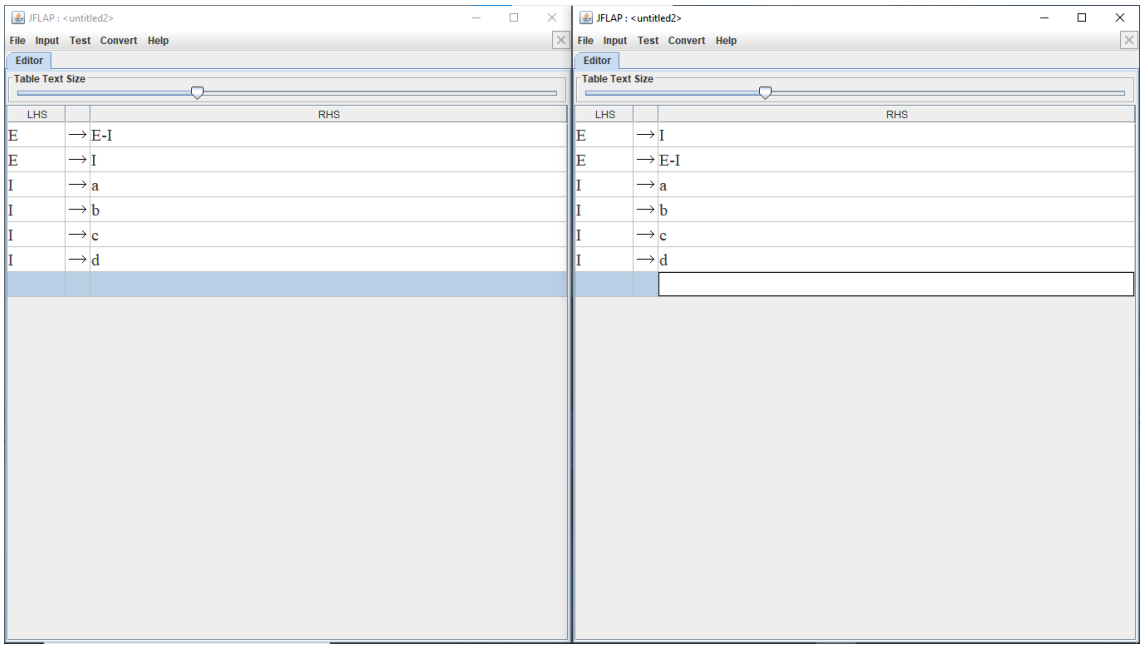
Para el primero, probamos a poner la gramática original en diferente orden:



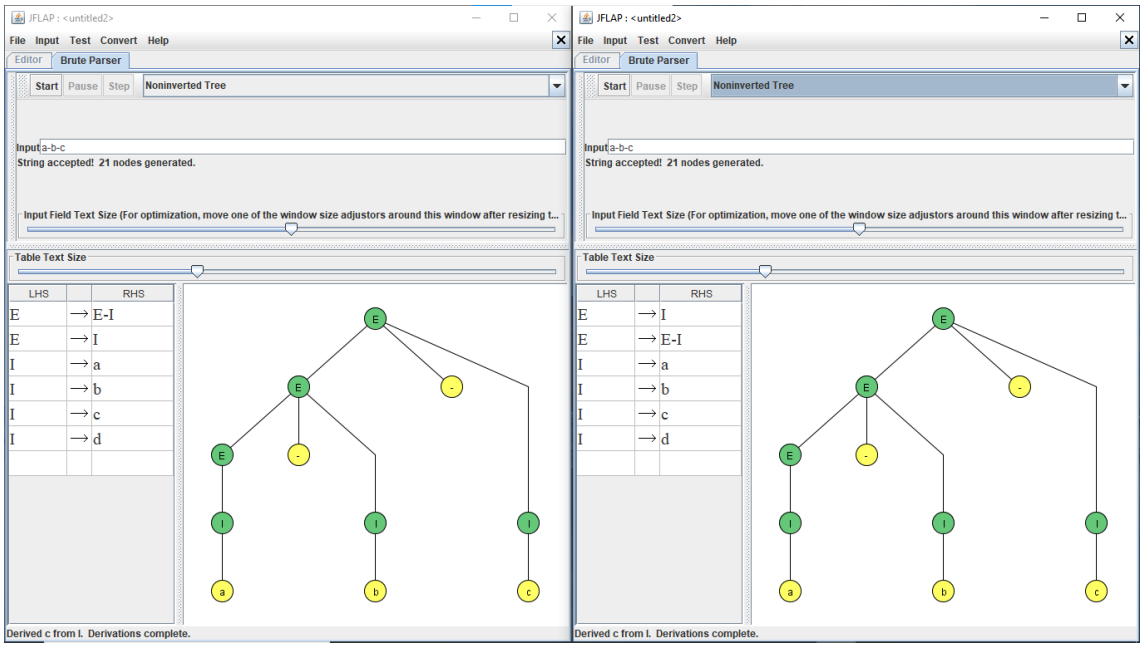
Y metemos la cadena “a – b – c”. Como veremos en la siguiente imagen, para una misma gramática (con el único cambio del orden) y para una misma cadena, los árboles de derivación son diferentes.



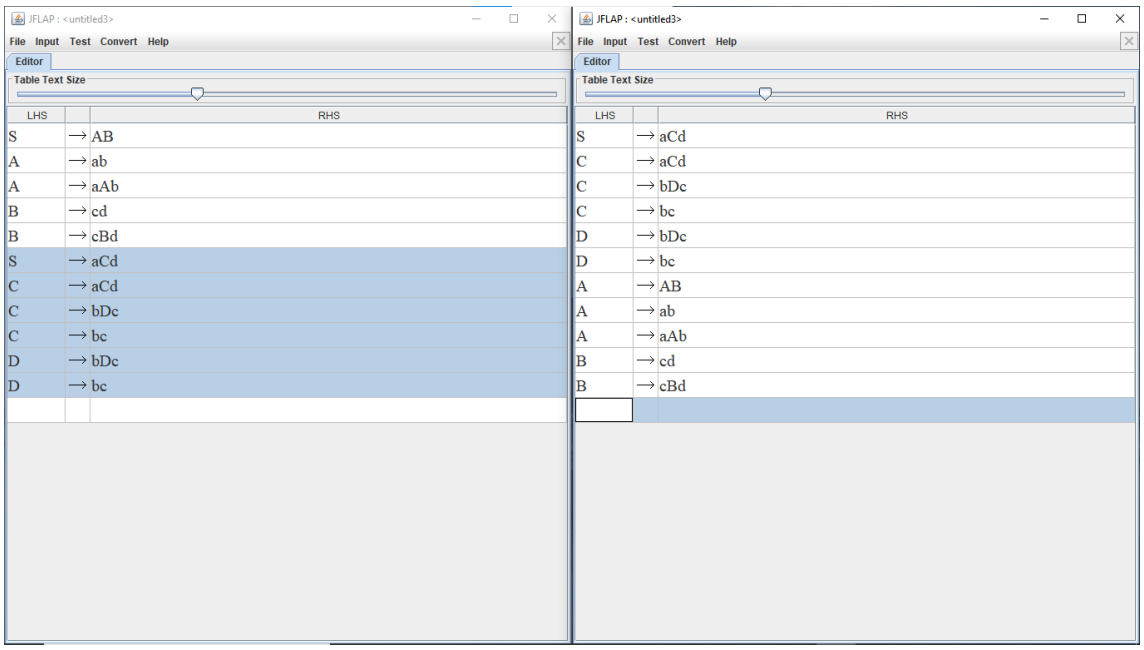
Pero si ahora quitamos la expresión E ⟶ I – E e igualmente cambiamos el orden del resto de expresiones, vemos que ha desaparecido la ambigüedad:



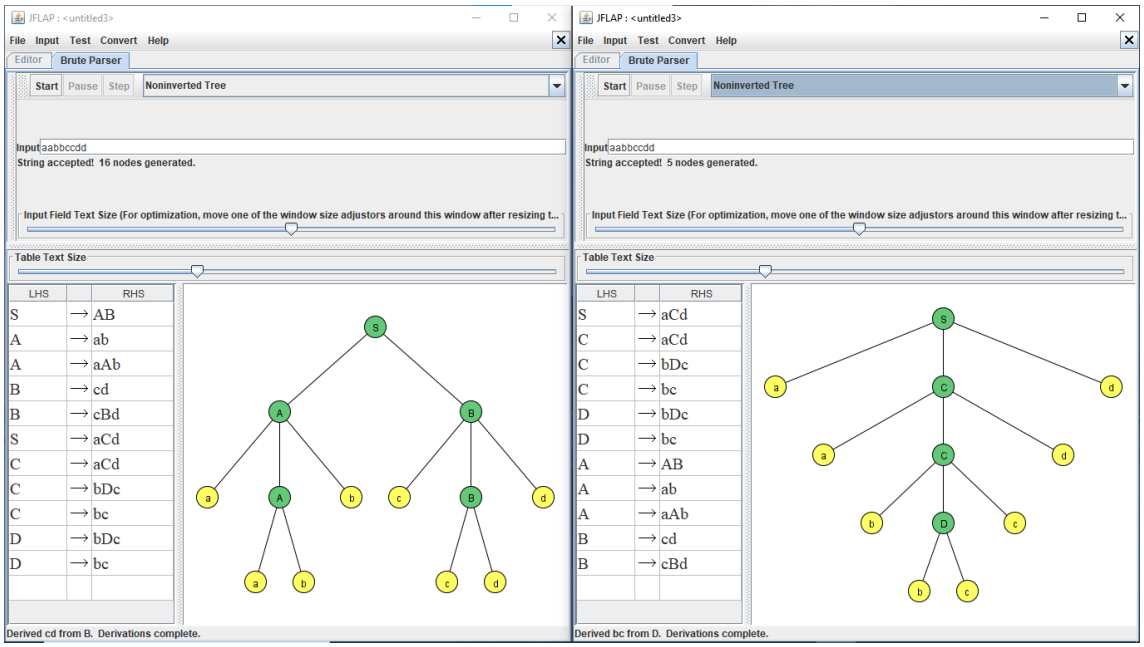
Y para la cadena “*a – b – c*” ya no hay ambigüedad.



Ahora pasamos al segundo lenguaje y vemos la ambigüedad de la misma manera que en el caso anterior, con la única diferencia es que aquí no podremos arreglarlo.



Y para la misma cadena "*aabbccdd*" vemos los árboles de derivación generados en cada caso:


JFLAP : <untitled3>
File Input Test Convert Help
Editor Brute Parser
Start Pause Step Noninverted Tree
Input aabbccdd
String accepted! 16 nodes generated.
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing t...
Table Text Size
LHS RHS
S → AB
A → ab
A → aAb
B → cd
B → cBd
S → aCd
C → aCd
C → bDc
C → bc
D → bDc
D → bc
Derived cd from B. Derivations complete.
JFLAP : <untitled3>
File Input Test Convert Help
Editor Brute Parser
Start Pause Step Noninverted Tree
Input aabbccdd
String accepted! 5 nodes generated.
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing t...
Table Text Size
LHS RHS
S → aCd
C → aCd
C → bDc
C → bc
D → bDc
D → bc
A → AB
A → ab
A → aAb
B → cd
B → cBd
Derived bc from D. Derivations complete.