**Capítulo 2, ejercicio 9**

Algorithm to Remove Left Recursion with an example:

Suppose we have a grammar which contains left recursion:

S-->S a / S b / c / d

Check if the given grammar contains left recursion, if present then separate the production and start working on it.

In our example,

S-->S a/ S b /c / d

Introduce a new nonterminal and write it at the last of every terminal. We produce a new nonterminal S’and write new production as,

S-->cS' / dS'

Write newly produced nonterminal in LHS and in RHS it can either produce or it can produce new production in which the terminals or non terminals which followed the previous LHS will be replaced by new nonterminal at last.

S'-->? / aS' / bS'

So after conversion the new equivalent production is

S-->cS' / dS'

S'-->? / aS' / bS'

Indirect Left Recursion:

A grammar is said to have indirect left recursion if, starting from any symbol of the grammar, it is possible to derive a string whose head is that symbol.

For example,

A --> Br

B --> Cd

C --> At

Where A, B, C are non-terminals and r, d, t are terminals.

Here, starting with A, we can derive A again on substituting C to B and B to A.

**Algorithm to remove Indirect Recursion with help of an example:**

A1 --> A2 A3

A2 --> A3 A1 / b

A3 --> A1 A1 / a

Where A1, A2, A3 are non terminals and a, b are terminals.

Identify the productions which can cause indirect left recursion. In our case,

A3--&gt A1 A1 / a

Substitute its production at the place the terminal is present in any other production substitute A1–&gt A2 A3 in production of A3. A3 –> A2 A3 A1.

Now in this production substitute A2–&gt A3 A1 / b and then replace this by,

A3 --> A3 A1 A3 A1 / b A3 A1

Now the new production is converted in form of direct left recursion, solve this by direct left recursion method.

Eliminating direct left recursion in the above,

A3 --> a | b A3 A1 | aA' | b A3 A1A'

A' --> A1 A3 A1 | A1 A3 A1A'

The resulting grammar is then:

A1 --> A2 A3

A2 --> A3 A1 | b

A3 --> a | b A3 A1 | aA' | b A3 A1A'

A' --> A1 A3 A1 | A1 A3 A1

**Capítulo 3, ejercicio 1**

Iniciamos con la gramatica (el paso 1 ya esta hecho): ´

• E → E + T|T ∗ F|(E)|a|b|Ia|Ib|I0|I1

• T → T ∗ E|(E)|a|b|Ia|Ib|I0|I1

• F → (E)|a|b|Ia|Ib|I0|I1

• I → a|b|Ia|Ib|I0|I1

Para el paso 2, introducimos nuevas variables y nos quedan

las siguientes reglas:

A → a, B → b, Z → 0, O → 1

P → +, M → ∗, L → (, R →)

y al reemplazar obtenemos la gramatica: ´

E → EPT|TMF|LER|a|b|IA|IB|IZ|IO

T → TPE|LEL|a|b|IA|IB|IZ|IO

F → LER|a|b|IA|IB|IZ|IO

I → a|b|IA|IB|IZ|IO

A → a, B → b, Z → 0, O → 1

P → +, M → ∗, L → (, R →

Para el paso 3, reemplazamos:

• E → EPT por E → EC1, C1 → PT

• E → TMF, T → TMF por

E → TC2, T → TC2, C2 → MF

• E → LER, T → LER, F → LER por

E → LC3, T → LC3, F → LC3, C3 → ER

La gramatica CNF final es: ´

• E → EC1|TC2|LC3|a|b|IA|IB|IZ|IO

• T → TC2|LC3|a|b|IA|IB|IZ|IO

• F → LC3|a|b|IA|IB|IZ|IO

• I → a|b|IA|IB|IZ|IO

• C1 → PT, C2 → MF, C3 → ER

• A → a, B → b, Z → 0, O → 1

• P → +, M → ∗, L → (, R →)