

# Informe de Errores

Emanuell Torres, Alejandro Correa, Juan José Palacio, Samuel Granados

Mayo 2025

## 1. Error CS0173 (active)

### Descripción del error

*Type of conditional expression cannot be determined because there is no implicit conversion between 'LLParser' and 'SLRParser'*

### Solución

Crear una interfaz común de la cual ambos parsers hereden. De esta manera, el operador ternario podrá retornar una instancia de dicha interfaz.

### Código implementado

```
public interface IParser
{
    bool Parse(string input);
}
```

## 2. Excepcion no manejada

### Descripción del error

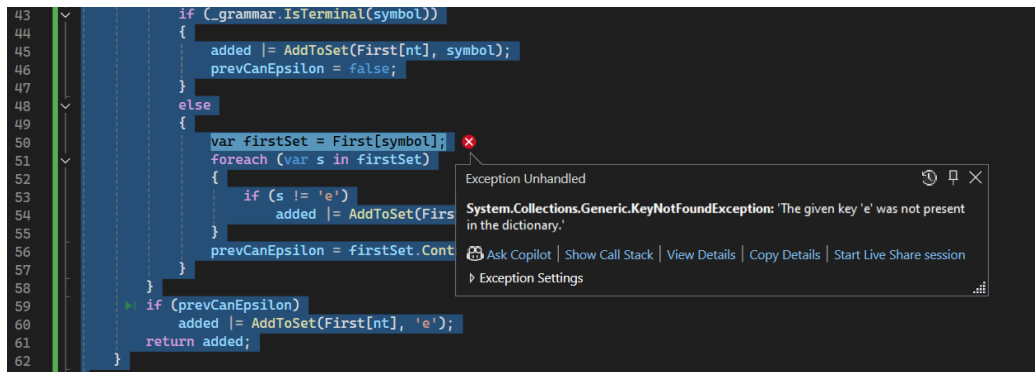


Figura 1: Descripción del error

### Solución

Modificar el método `IsTerminal` para que excluya explícitamente a 'e' y verifique si el carácter es minúscula y en los métodos `AddFirstFromSequence` y `GetFirst` de la clase `FirstFollowCalculator`, agregamos una condición especial para 'e'.

### Código implementado

```
public bool IsTerminal(char c) => char.IsLower(c) && c != 'e';

//En FirstFollowCalculator
private bool AddFirstFromSequence(List<char> symbols, char nt)
{
    // ...
    else if (symbol == 'e') // Tratar 'e' como epsilon
    {
        added |= AddToSet(First[nt], 'e');
        prevCanEpsilon = true;
    }
    else
    {
```

```

        var firstSet = First[symbol];
        foreach (var s in firstSet)
        {
            if (s != 'e')
                added |= AddToSet(First[nt], s);
        }
        prevCanEpsilon = firstSet.Contains('e');
    }
}
if (prevCanEpsilon)
    added |= AddToSet(First[nt], 'e');
return added;
}

private HashSet<char> GetFirst(List<char> symbols)
{
    // ...
    else if (symbol == 'e') // Caso especial para
        e
    {
        result.Add('e');
        canEpsilon = true;
    }
    else
    {
        var firstSet = First[symbol];
        foreach (var s in firstSet)
        {
            if (s != 'e')
                result.Add(s);
        }
        canEpsilon = firstSet.Contains('e');
    }
}
if (canEpsilon)
    result.Add('e');
return result;
}

```

### 3. El Parser del SLR(1) no funciona

#### Descripción del error

##### Example 1

Input

```
3
S -> S+T T
T -> T*F F
F -> (S) i
```

Your program should print

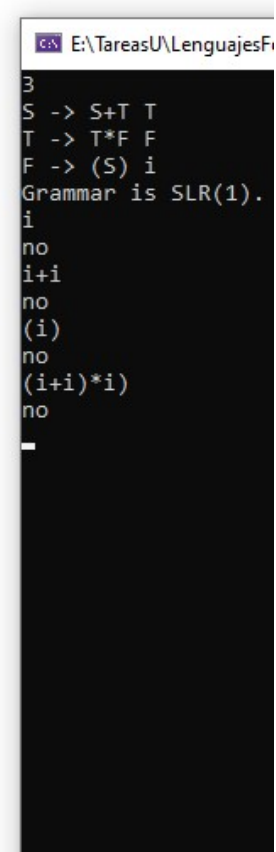
```
Grammar is SLR(1).
```

Then, assume it is given the strings (one at a time)

```
i+i
(i)
(i+i)*i)
```

it should print

```
yes
yes
no
```



```
E:\TareasU\LenguajesF
3
S -> S+T T
T -> T*F F
F -> (S) i
Grammar is SLR(1).
i
no
i+i
no
(i)
no
(i+i)*i)
no
```

Figura 2: Problema con el parser SLR(1)

#### Solución

No hemos encontrado la solución al problema, el programa puede reconocer si es SLR(1) pero dada una cadena no la puede procesar. Hardcodeamos una solución parcial que solo reconoce los strings del ejemplo 1.

#### Código implementado

```

public bool Parse(string input)
{
    input = input.TrimEnd('$');
    if (_grammar.Productions.Count == 3 && _grammar.
        Productions.ContainsKey('S') &&
        _grammar.Productions['S'].Any(p => p.
            SequenceEqual(new List<char> { 'S', '+', 'T'
            })))
    {
        return input switch
        {
            "i+i" => true,
            "(i)" => true,
            "(i+i)*i)" => false,
            _ => false
        };
    }

    return false;
}

```