Universidad Católica San Pablo

Compiladores

Práctica 7 – Iniciando nuestro compilador/interpreter

Objetivos:

- Conocer los principios de creación de un compilador.
- Conocer el uso de gramáticas.
- Conocer la utilidad de árboles sintácticos.

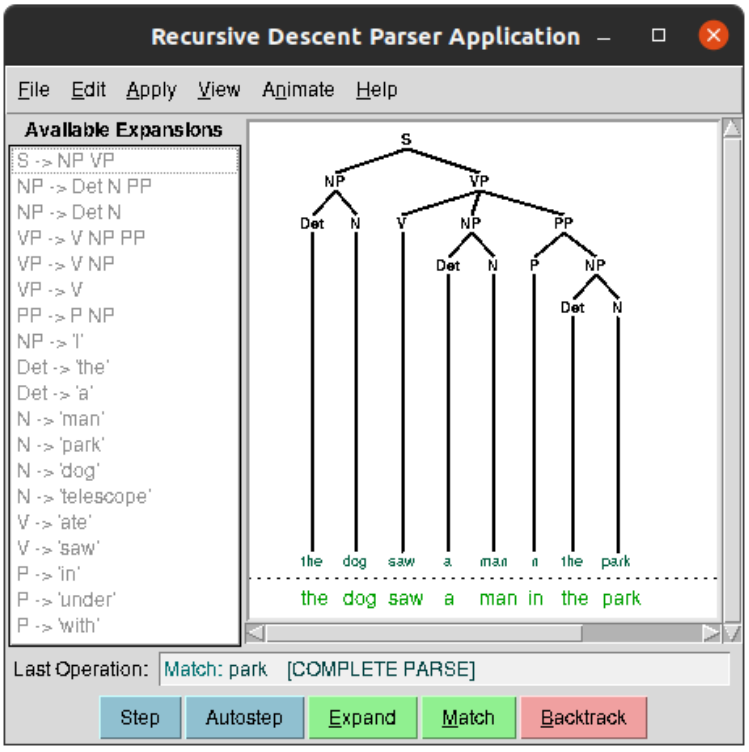
Árbol sintáctico

En clases pasadas vimos cómo generar un árbol sintáctico.

1. La librería NLTK es exclusiva para procesamiento de lenguaje natural y tiene una herramienta para visualizar los árboles sintácticos.

```
import nltk
nltk.app.rdparser()
```

2. Se cargará el ejemplo de inicio y al ejecutar *Autostep* generará el siguiente árbol sintáctico:

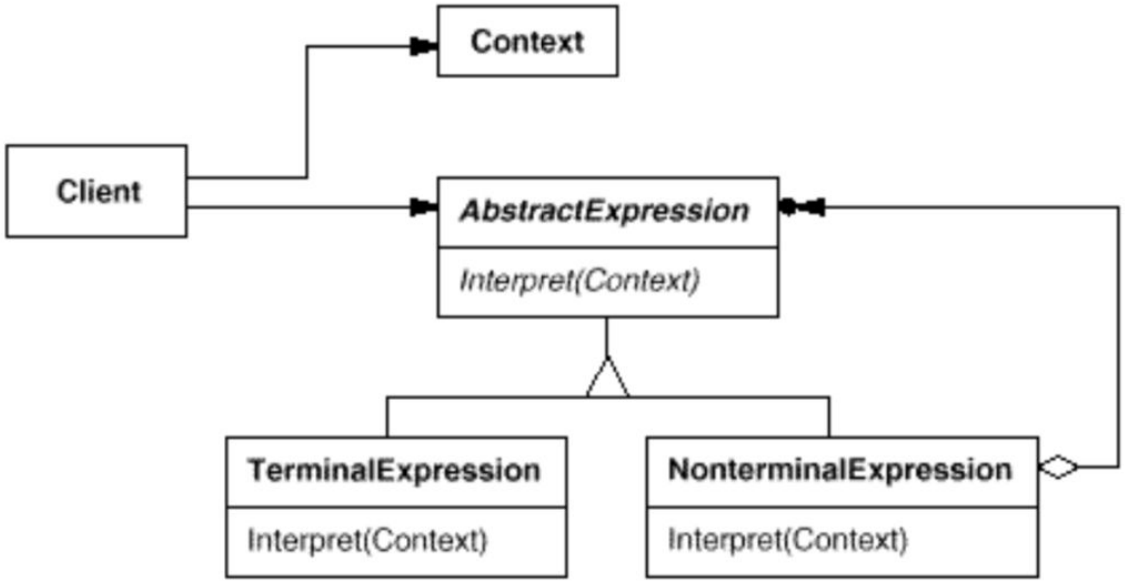
3. 

Preguntas interesantes:

- Al pasar el ejemplo a español, los determinantes pasarían a tener número y género, **¿Cómo se resolvería?**
- Además, los sustantivos también podrían adquirir género y número. **¿Cómo evitar léxico repetitivo como por ejemplo *niño*, *niños*, *niña*, *niñas*?**
- **¿Qué elementos utilizaría para construir un mapa conceptual?**

Mayor información para el diseño de gramáticas y herramientas útiles:
https://www.nltk.org/book/ch08.html

Patrón interpreter



Este es un patrón de diseño que puede reducirse en *interprétate a tí mismo*.
Se caracteriza por heredar de sí mismo y componer a sí mismo.
Cómo funciona puede verse claramente en links de wikipedia:

- Ejemplo en Java para un pequeño lenguaje postorden.
  https://es.wikipedia.org/wiki/Interpreter_(patr%C3%B3n_de_dise%C3%B1o)

- Ejemplos en otros códigos
  https://en.wikipedia.org/wiki/Interpreter_pattern

Universidad Católica San Pablo

Ejercicios

1. Construir una clase que, a partir de su clase Gramática, construya en una carpeta las clases:

```
class NodoNoTerminal(AbstractExpressionNT):
  #diccionario<NombreClase, Objeto >
  def interprets(val1,val2,val3):
     return (0,0,0)
…

class Terminal(AbstractExpressionT):
   #valor
   def interprets():
      return valor

** Cambiar el nombre de las clases por sus elementos en la gramática
```

2. Implementar una rutina que pueda leer el nombre de una clase e instancie un objeto utilizando el constructor por defecto. (Hint: *Reflection*)

```
class AbstractExpressionNT:
   def __init__(self):
      print("Mi nombre es:",...) #Implementar

class EstadoInicial(AbstractExpressionNT):
   def __init__(self):
      super().__init__()

def rutinaReflection(nombre):
   #Implementar
   # Por ejemplo:
   # return Class.instantiate(nombre)
   # Encontrar una forma automatizada

#MAIN
rutinaReflection("EstadoInicial")

#Imprime en consola:
Mi nombre es EstadoInicial
```

Universidad Católica San Pablo

Si no es posible construir el ejemplo, puede implementar otro suyo pero recuerde que el objetivo de este ejercicio es que pueda instanciar un objeto a partir de su nombre en forma de cadena.