TEC | Tecnológico de Costa Rica

Curso: Lenguajes, Compiladores e Interpretes

Profesor: Marco Rivera Meneses

Tarea Programada 2

Alumnos:

García Downing Geovanny, 202009224

Martínez Vargas Andrés, 2019061822

Mejías Hernández Diana, 2020077281

II Semestre

2022

Tabla de contenido

Descripción de las reglas implementadas:

Reglas de oración/sentence:

Para poder construir y validar que exista una oración en español (oracion) o en inglés (sentence) se deben seguir una serie de reglas y criterios previamente establecidos de los cuales podemos tomar como ejemplo dos de cada uno de ellos:

```
oracion(s(S,V)) --> sintagma_nominal(S,PERS,NUM), verbo(V,PERS,NUM).
oracion(s(S)) --> saludos(S).
```

```
sentence(s(S,V)) --> nominal_predicate(S,NUM), verb(V,NUM).
sentence(s(S)) --> grettings(S).
```

Como estas reglas, existen siete formas diferentes de construir una oración en cualquiera de los dos lenguajes.

Al ser una gramática libre de contexto, se deben remplazar los no terminales con terminales (reglas) al finalizar el recorrido por la gramática. De este modo, el sintagma nominal y el verbo se puede remplazar respectivamente con, por ejemplo, “Messi” y “Jugar” para formar la oración “Messi juega”.

```
sujeto(suj(su21), _, tercera, singular) --> ["messi"].
verbo(v(v25), tercera, singular) --> ["juega"]. % El / Ella
```

Las combinaciones posibles que se pueden realizar al tener 20 verbos destinitos y 50 sujetos/sustantivos diferentes, es alrededor de 1.2×10^{20} combinaciones distintas, esto sin contar artículos, preposiciones, adjetivos y distintas formas del mismo verbo.

Descripción de las estructuras de datos desarrolladas:

Estructura del lenguaje común

Para poder hacer la traducción de un lenguaje a otro, se decidió hacer un lenguaje en común que tanto español como ingles pudieran entender, y transformarse en él. Este lenguaje en común resulta ser un lista de listas. Dentro de la lista se encuentra la oración original descompuesta en terminales.

Para entender el concepto de este lenguaje común y como es una estructura de listas dentro de listas, tomaremos el ejemplo de la oración “ella va hacia la escuela”. Para este caso, el lenguaje en común (A) devolvería la siguiente estructura:

```
?- phrase(oracion(A), ["ella", "va", "hacia", "la", "escuela"]).
A = s(det(suj(su5)), v(v9), det(p(p1), a(art), sus(s12)))
```

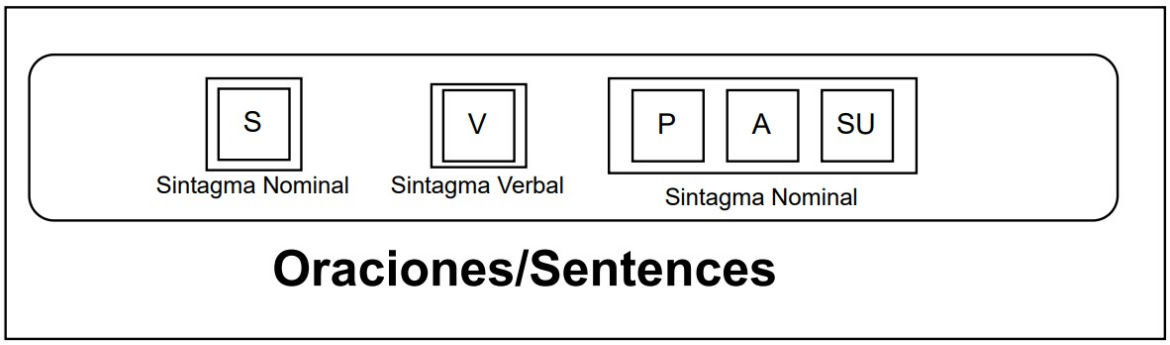

Figura 1. Estructura de datos del lenguaje en común

Descripción de los algoritmos desarrollados:

Algoritmo backtracking:

En prolog, el backtracking se hace por defecto. Esto facilita desarrollar una gramática libre de contexto.

En el caso de este proyecto, este algoritmo se utilizó para determinar el lenguaje en común entre inglés y español, ya que por ejemplo, para que la oración “badbunny tiene diamantes” tenga lógica y cumpla con ser una oración valida, se debe determinar cuál de las diferentes formas del verbo “tener” se tiene, en este caso está en tercera persona singular. Esta información coincide con la del sujeto “badbunny”, ya que se podría remplazar “badbunny” por “el” (tercera persona singular). De esta manera, se valida y se garantiza la existencia de una oración de la misma naturaleza al convertirla en inglés.

Problemas sin solución:

El desarrollo de no solo oraciones, sino párrafos completos no pudo ser incorporado en el código final, ya que se pudo definir el estilo y las reglas de una oración, mas no la de un párrafo completo.

Los signos de puntuación son un faltante en la versión final del programa, esto debido a que la separación que se hace en la lista de lenguaje intermedio se toma con respecto a los espacios entre cada palabra, y dado que los signos de puntuación van sin espacios inmediatamente después de finalizar una oración en inglés o al inicio y final en el caso del español, fue imposible determinar cuando existe un signo de puntuación.

Actividades realizadas por estudiantes:

	Integrantes		
	Geovanny García Downing	Andrés Martínez Vargas	Diana Mejías Hernández
Semana #1	Investigación sobre el paradigma lógico en Prolog Investigación sobre la librería “phrase” Inicio de la documentación externa del proyecto Reunión para dividir tareas puntuales y resolver de problemas específicos	Investigación sobre el paradigma lógico en Prolog Investigación sobre la librería “phrase” Inicio de la documentación externa del proyecto Inicio de la bitácora del proyecto Reunión para dividir tareas puntuales y resolver de problemas específicos	Investigación sobre el paradigma lógico en Prolog Investigación sobre la librería “phrase” Inicio de la documentación externa del proyecto Reunión para dividir tareas puntuales y resolver de problemas específicos
Semana #2	Implementación de una interfaz más amigable y continua al iniciar el programa Corrección de errores Implementación de preguntas en el lenguaje	Implementación de una base de datos más poblada, con al menos 20 verbos y 50 sujetos/sustantivos Finalización de la documentación externa del proyecto	Implementación de la lógica de la traducción Implementación de los saludos en el lenguaje Corrección de errores

	Finalización de la documentación externa del proyecto Reunión para probar la funcionalidad del código	Finalización de la bitácora del proyecto Reunión para probar la funcionalidad del código	Finalización de la documentación externa del proyecto Reunión para probar la funcionalidad del código

Problemas encontrados:

Integración con el usuario:

El código no tiene la función de detectar que lenguaje se digita ni a cual se desea traducir, esto implica que si no se establece esto con anterioridad, no dará ningún resultado si la traducción la toma al revés.

a) **Intentos de solución**

- Detectar el lenguaje.
- Se decidió implementar una función en la cual se pregunta desde un inicio al usuario que idioma desea elegir.

b) **Solución encontrada**

- Preguntar desde el inicio a que idioma se desea traducir resultó ser la mejor opción, de este modo se pudo tener un mayor control de la traducción desde el inicio.

c) **Recomendaciones**

- La solución más sencilla suele ser la mejor, por lo que si no se pido lo contrario, preguntar al usuario que desea es lo mejor

d) **Conclusiones**

- Preguntar al usuario la opción a elegir

Continuidad del programa:

El programa desarrollado tiene como objetivo la traducción de una oración. Sin embargo, no se pensó en un sistema que se pueda utilizar una vez termina la traducción, es decir, el programa termina inmediatamente después de realizar la traducción.

a) **Intentos de solución**

- Hacer el programa recursivo.

b) **Solución encontrada**

- Hacer una función recursiva que permita traducir cuantas veces se desee hasta que el usuario decida lo contrario.

c) **Recomendaciones**

- El programa no debería terminar sin que el usuario lo decida primero.

d) **Conclusiones**

- Hacer una función recursiva para traducir cuantas veces se desee.

Orden de las oraciones en cada lenguaje:

Uno de los mayores desafíos en la traducción fue el orden lógico que sigue cada lenguaje, pues en español es común ver adjetivos seguidos de un sustantivo, como por ejemplo “el perro pequeño” y en inglés es al revés, primero el sustantivo y después el adjetivo, siguiendo el ejemplo anterior “the small dog”.

a) **Intentos de solución**

- Cambiar el orden de los terminales o reglas para formar una oración en inglés (sentence), más no en el árbol de generación de una oración para el lenguaje en común entre inglés y español..

b) **Solución encontrada**

- Alterar el orden de los terminales y no la del árbol gramatical.

c) **Recomendaciones**

- Al solamente mover de posición el orden de las reglas gramaticales en el código, se ahorra mucho código empleado en analizar únicamente e individualmente un lenguaje.

d) **Conclusiones**

- Reordenar el código para abarcar la mayor cantidad de lenguajes posibles.

Existencia de homógrafos:

Este resultó ser un problema muy grande para este proyecto, ya que para poder traducir los homógrafos (palabras que se escriben igual, pero tienen significados diferentes) se debe entender el contexto de la oración y esta área esta fuera de los alcances de este proyecto.

a) **Intentos de solución**

- Entender el contexto de una oración
- No poner homógrafos.

b) **Solución encontrada**

- La mejor solución claramente fue no incluirlos en el proyecto y buscar sustantivos que no pusieran malinterpretarse.

c) **Recomendaciones**

- No meterse en casos especiales que puedan facilitar la creación de bugs.

d) **Conclusiones**

- Hacer el código simple y sin tantas excepciones facilita la legibilidad del código y elimina la posibilidad de bugs.

Conclusiones y recomendaciones

Prolog permite consultar fácilmente hechos y reglas haciendo backtracking. Esto es muy beneficiosos si lo que se pretende hacer es encontrar casos específicos a un conjunto enorme de reglas y hechos.

La librería “phrase” facilita mucho la tarea para de ordenar en una lista los nodos que se crean en un árbol. En nuestro caso, toma el árbol gramatical y crea una lista con las palabras ya traducidas.

No se puede hacer una traducción literal de las palabras, ya que esto conduciría a errores sintácticos o semánticos, como lo es traducir “el libro rojo” a “the book red”. Además una traducción real, debe saber de un modo u otro el contexto con el que se dice una oración, ya que podrían ocurrir errores como traducir “she can” a “ella lata”. Por último, un problema que aun los mejores traductores experimentan es traducir incluyendo diferentes géneros en español, por ejemplo, “the nurse” se traduciría como “el enfermero” o “la enfermera”, no existe una sola solución para este caso.

Se deben conocer y tomar en cuenta las frases y estilos de habla pertenecientes a cada lenguaje para que tenga sentido la traducción, un ejemplo de esto es que, aunque la frase “i swing and i miss” se traduzca correctamente a “me balanceo y extraño”, carece de sentido, pues realmente lo que se intenta decir es “golpeo y fallo”. Una frase más cercana a esta idea es traducir “pura vida” a “pure life”, como podemos ver, carece de sentido. Por lo que estos casos hay que tomarlos como casos separados en la traducción y no intentar traducirlos como en cualquier otro caso.

Bitácora:

Fecha: 22 de octubre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías
- Geovanny García

Actividades realizadas:

- Se conversó sobre el proyecto y se estableció´ una estrategia de desarrollo. Se asignó que Andrés poblara la base de datos y tuviera control de la bitácora. Diana se le asignó la tarea de controlar la lógica detrás de las traducciones y Geovanny se le encargo los casos especiales de la traducción como lo son los saludo y puntos de puntuación.
- Se creó un repositorio para controlar el historial de versiones del código.

Fecha: 23 de octubre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías
- Geovanny García

Actividades realizadas:

- Se inició una investigación por parte de los tres participantes acerca del paradigma lógico en Prolog, así como una búsqueda de información acerca de la librería “phrase”.

Fecha: 25 de octubre de 2022

Responsable(s):

- Diana Mejías

Actividades realizadas:

- Se tiene un versión inicial sobre la traducción de oraciones completas, se consideran nuevos casos y se toma nota de las deficiencias para generar una mejor versión.

Fecha: 27 de octubre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías
- Geovanny García

Actividades realizadas:

- Geovanny introduce los primeros saludos y frases especiales como parte de la lista de oraciones que se pueden traducir.
- Se reúne el equipo para comunicar lo que ya se tiene y lo que hace falta, se dan diferentes puntos de vista para los problemas que se encuentran, se asignan pequeñas tareas para avanzar con el proyecto y se consulta con el profesor dudas que surgieron a en la misma reunión con respecto a la funcionalidad del código

Fecha: 29 de octubre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías

Actividades realizadas:

- Diana agrega las preguntas a las posibilidades de oraciones que se pueden traducir.
- Andrés pobla la base de datos para que esta cuente con 50 sujetos o sustantivos y 20 verbos. Además documenta en código interno para que este sea más legible.

Fecha: 31 de octubre de 2022

Responsable(s):

- Diana Mejías

Actividades realizadas:

- Diana incorpora los sinónimos al código para que estos se puedan mostrar a la hora de traducir diferentes palabras u oraciones.

Fecha: 01 de noviembre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías
- Geovanny García

Actividades realizadas:

- El grupo nuevamente se reúne para poder ver lo que se ha logrado hasta el momento. Se comentan la rúbrica de evaluación para poder enfocar el proyecto al resultado esperado por el profesor.
- Finaliza el día con el código listo y en completo funcionamiento. Todos los aspectos previamente considerados ahora están en el código.

Fecha: 02 de noviembre de 2022

Responsable(s):

- Andrés Martínez
- Diana Mejías
- Geovanny García

Actividades realizadas:

- Andrés revisa, termina y da formato a la documentación. Además, finaliza la bitácora sobre las actividades realizada a lo largo del tiempo del proyecto. Por último, agrega los problemas sin solución, que previamente no pudieron ser considerados.
- Geovanny crea el manual de usuario para poder guiar al usuario a usar correctamente el programa.
- El grupo se reúne para dejar claro como quedó finamente el código.