Inteligencia Artificial

Proyecto del curso
Mayo 26 del 2015
Carlos Fernando Motta

Problema

El proyecto de curso consiste en diseñar e implementar un programa en lisp que permita hacer razonamiento hacia adelante (Forward Chaining) y razonamiento hacia atrás (Backward Chaining) sobre una base de conocimiento que debíamos diseñar de acuerdo a las reglas establecidas en el taller 2. Adicionalmente, la base de conocimientos debía ser de relaciones familiares, donde se debían incluir los siguientes predicados: padre, madre, hijo, hija, primo, prima, primo segundo, prima segunda, t´ıo, t´ıa, sobrino, sobrina, pariente, suegro, suegra, yerno, nuera, cuñado, cuñada.

Diseño de la solución

Según la definición del problema, el diseño de la solución de este proyecto se divide en dos partes, la creación del algoritmo de búsqueda hacia adelante (*Forward Chaining*), la creación del algoritmo de búsqueda hacia atrás (*Backward Chaining*) y la creación de la base de conocimiento (*Knowledge Base*) que represente algunas relaciones del árbol genealógico. En las secciones siguientes se explica en más detalle cada uno de estos diseños y las limitaciones presentes del sistema.

Para la solución del proyecto se asume lo siguiente:

- Las variables van en minúsculas.
- Los nombres de predicados empiezan por P mayúscula.
- Las constantes van en mayúsculas y no pueden tener P como la primera letra.
- Las reglas tienen el siguiente formato: (((antecedente) (antecedente) ...) (consecuente)).

Forward Chaining

En esta parte del proyecto se decidió tomar como base la implementación vista en clase y el algoritmo de forward-chaining explicado en [2].

El algoritmo va construyendo el árbol de la figura 2 en la medida que avanza en la recursión para el caso ejemplo enviado por la profesora en el segundo taller, el cual es mismo caso trabajado en el libro de Norvig en [2].

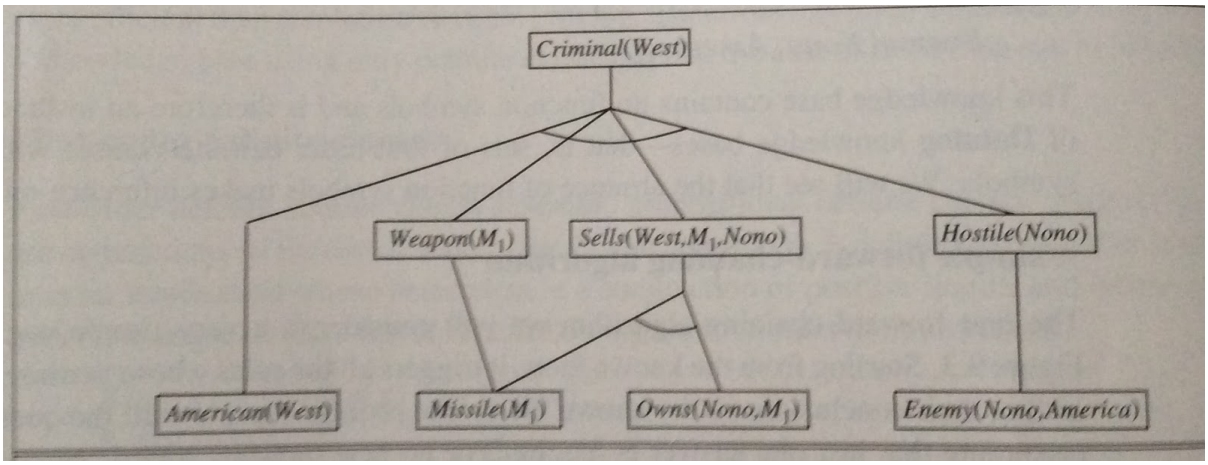

Figura 1. Árbol generado por el Forward-Chaining para el ejemplo de Criminal(west)

Ahora bien, la idea de este algoritmo consiste en inferir un *query* o una pregunta que se le hace al sistema partiendo de los hechos que se conocen e ir iterando sobre dicha base de conocimientos hasta lograr responder Verdadero si se logra inferir y Falso en el caso contrario.

Así bien, como ya se había explicado en el informe del taller 2, el algoritmo utiliza tres funciones principales para hacer el razonamiento, **unify**, **unPaso** y **forward**. La primera función se encarga de hacer unificaciones, la segunda se encarga de iterar sobre la base de conocimientos, hacer inferencias en ese nivel de la iteración y adicionar los nuevos hechos a la base de conocimiento y finalmente, la tercera función se encarga de decidir si se pudo sacar nuevos hechos en cada paso de la iteración que se da, y así en el caso en el que no se pueda inferir más, valida el query con lo que se pudo deducir y decide Verdadero o Falso, y en el caso de que todavía pueda inferir más, esta función llama a la función **unPaso** para seguir intentando inferir nuevos hechos.

A diferencia de la entrega realizada el 12 de Abril en el segundo taller, en el proyecto se le corrigieron varios aspectos al algoritmo, el principal y más importante de todos fue la capacidad de unificar una variable y propagarse sobre esa regla para lograr saber si era posible probar los antecedentes de dicha regla, pues en la entrega pasada el algoritmo sólo validaba que pudiera probar cada antecedente con los hechos existentes sin importar si era la misma variable o no.

<u>Limitaciones</u>

El algoritmo no logra identificar más de una posible unificación, es decir, si la base conocimiento tiene más de una posible unificación para cada uno de los antecedentes de una regla de la base de conocimiento, el algoritmo toma únicamente el primero que pueda deducir.

Presenta un problemas al momento de unificar predicados que tenga más de una variable.

Backward Chaining

En esta parte del proyecto se decidió tomar como base la implementación vista en clase y el algoritmo de backward-chaining explicado en [2].

El algoritmo va construyendo el árbol de la figura 2 en la medida que avanza en la recursión para el caso ejemplo enviado por la profesora en el segundo taller, el cual es mismo caso trabajado en el libro de Norvig en [2].

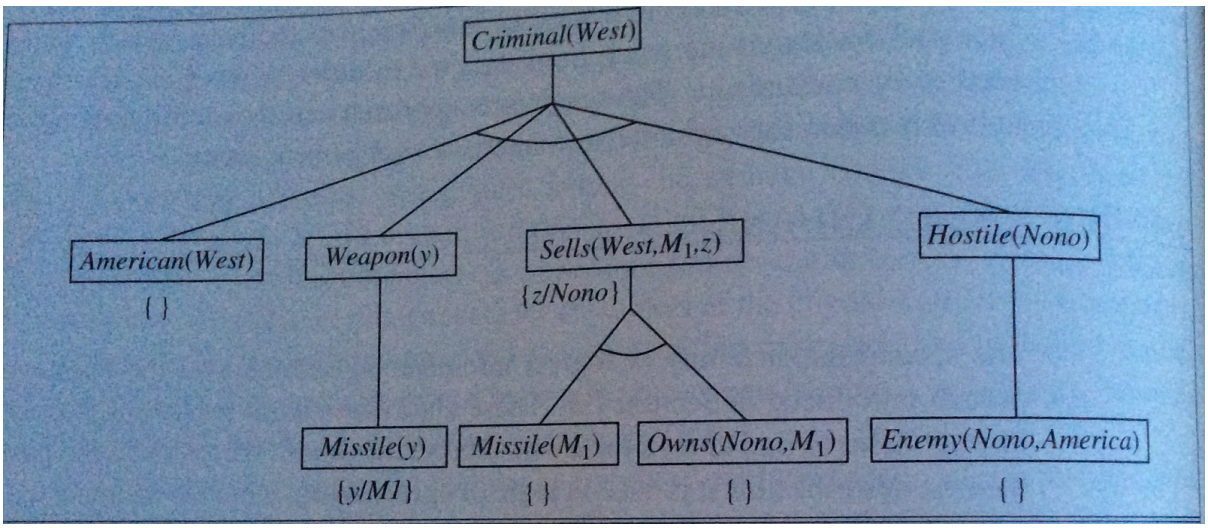

Figura 2. Árbol generado por el Backward-Chaining para el ejemplo de Criminal(west)

Ahora bien, la idea de este algoritmo consiste en deducir un *query* o una pregunta que se le hace al sistema partiendo dicho *query* se evalúa si se puede unificar alguno de los antecedentes, si alguno unifica entonces se procederá a deducir cada antecedente hasta llegar a los hechos y si en cada caso se puede unificar algo, el algoritmo puede entonces responder Verdadero si logra deducir y Falso en el caso contrario.

La estructura básica del algoritmo consta de cuatro funciones, **unify**, **unPasoAtras**, **propagate** y **backward**. La primera función es la misma que la utilizada en el algoritmo forward, la cual se encarga de hacer unificaciones. La segunda cumple con una función equivalente a la función **unPaso** en el algoritmo forward, sin embargo esta función busca iterar sobre la base conocimiento hasta lograr identificar el consecuente que puede unificar con el *query* actual para así poder propagar los antecedentes. La tercera función cumple, como su nombre lo indica, con propagar los antecedentes, es decir con lograr probar cada antecedente enviado desde la función **unPasoAtras**. Finalmente, la función backward es la que une las tres funciones anteriores al identificar si se puede unificar algo o no para así devolver False si no se puede y de lo contrario llamar a la función **unPasoAtras** para que se encargue de hacer una nueva iteración hasta lograr devolver Verdadero o False en caso contrario.

Limitaciones

Presenta un problemas al momento de unificar predicados que tenga más de una variable.

Debido a un problema en la función de unificación del algoritmo, ciertos queries no son probados como es debido y por lo tanto algunos resultados no son correctos.

Base de Conocimiento

La base de conocimiento fue construida utilizando como base la siguiente figura.

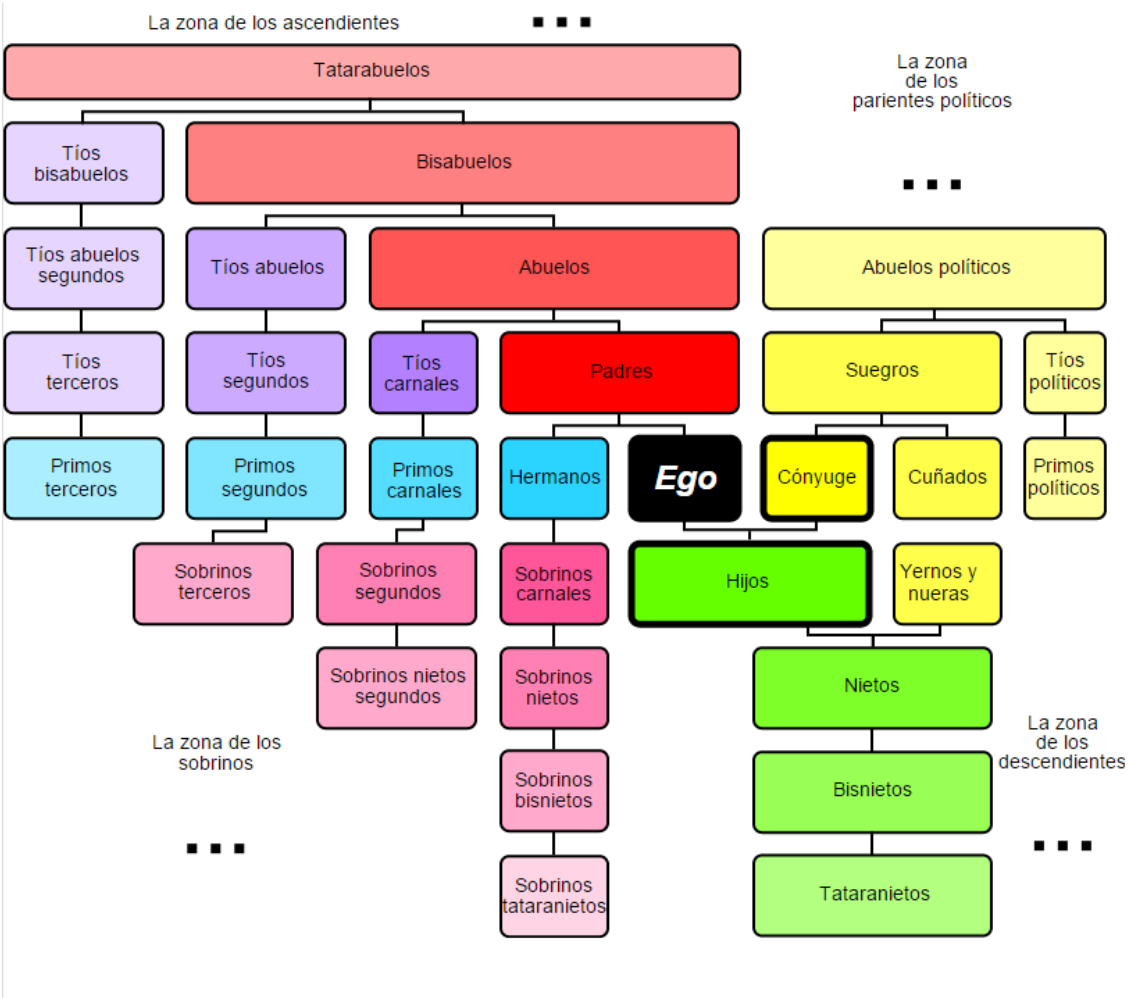

Figura 3. Estructura de un Árbol Genealógico

Limitaciones

No se realizaron suficientes pruebas y por lo tanto existe la posibilidad de que se presenten inconsistencias al momento de probar ciertas *queries.*

Pruebas

Las pruebas fueron realizadas utilizando las siguientes bases de conocimiento:

```
(setq kb '(  (((PAmerican x)(PWeapon y)(PSells x y z)(PHostile z)) (PCriminal x))
             ((()(POwns Nono M1))
             ((()(PMissile M1))
             (((PMissile x)(POwns Nono x)) (PSells West x Nono))
             (((PMissile x)) (PWeapon x))
             (((PEnemy x America)) (PHostile x))
             ((()(PAmerican West))
             ((()(PEnemy Nono America)) ))

(setq query '(PCriminal West))
```

Figura 4. Knowledge base 1.

```
(setq query2 '(PEvil Motz))

(setq kb3 '((((PKing x)(PGreedy x)) (PEvil x))
 ((()(PKing John))
 ((()(PGreedy Jenny))
 ((()(PKing Jenny))
 ((()(PGreedy John))
 ((()(PGreedy Joan))
 ((()(PKing Juan)) ))

(setq query3 '(PEvil John))
(setq query4 '(PEvil Jenny))
(setq query5 '(PEvil Joan))
(setq query6 '(PEvil Juan))
```

Figura 5. Knowledge base 2.

```
(setq kbp '( (((PHombre x)(PMadre z x)(PMadre z y)(PNIgual x y))(PHermano x y))
             ; (((PHombre x)(PPadre y x)(PMadre z x))(PHijo x y))
              (((PHombre x)(PPadre y x))(PHijo x y))
              (((PMujer x)(PPadre y x))(PHija x y))
              (()(PMadre Luisa Carlos))
              (()(PMadre Luisa Camilo))
              (((PHombre x)(PMadre y x)(PTia y z))(PPrimo x z))
              (((PMujer x)(PMadre y x)(PTia y z))(PPrima x z))
              (((PMujer x)(PMadre y z)(PHermana x y))(PTia y z))
              (((PHombre x)(PHijo x y)(PHijo y z)(PHermano z w)(PPadre w v)(PPadre v c))(PPrimoDos x c))
              (((PHombre x)(PHijo x y)(PHermano y z))(PSobrino x z))
              (((PMujer x)(PHija x y)(PHermano y z))(PSobrina x z))
              (((PHombre x)(PPadre x y)(PConyuge y z))(PSuegro x z))
              (((PMujer x)(PMadre x y)(PConyuge y z))(PSuegra x z))
              (((PHombre x)(PConyuge x y)(PHija y z))(PYerno x z))
              (((PMujer x)(PConyuge x y)(PHijo y z))(PNuera x z))
              (((PHombre x)(PHermano x y)(PConyuge y z))(PCunado x z))
              (((PMujer x)(PHermana x y)(PConyuge y z))(PCunada x z))
              (((PHombre x)(PHermano x y)(PPadre y z))(PPariente x z))
              (()(PHombre Carlos))
              (()(PHombre Camilo))
              (()(PMujer Luisa))
              (()(PMujer Stephany))
              (()(PHombre Teo))
              (()((PPadre Teo Carlos)))
              (()(PHermano Hugo Maria))
             (()(PMadre Luisa Juan))
             (()(PPadre Hugo Juan))
             (()(PTia Maria Juan))
             (()(PPadre Juan Clara))
             (()(PMadre Elsa Clara))
             (()(PMadre Clara Bruno))
             (()(PMadre Maria Jesus))
             (()(PPadre Jose Jesus))
             (()(PTio Hugo de Jesus))
             (()(PPadre Jesus Jorge))
             (()(PMadre Ana Jorge))
             (()(PPadre Jorge Bruno))
             (()(PNuera Elsa Luisa))
             (()(PNuera Elsa Hugo))
             (()(PNuera Ana Maria))
```

Figura 6. Knowledge base 3

Las pruebas del algoritmo forward sobre la base de conocimiento de la figura 4 se muestran en la figura 7, las pruebas sobre la base de conocimiento de la figura 5 se muestran en la figura 8 y las pruebas sobre la base de conocimiento de la figura 6 se muestran en la figura 9.

Las pruebas del algoritmo backward sobre la base de conocimiento de la figura 4 se muestran en la figura 10, las pruebas sobre la base de conocimiento de la figura 5 se muestran en la figura 11 y las pruebas sobre la base de conocimiento de la figura 6 se muestran en la figura 12.

```
Type :h and hit Enter for context help.

[1]> (load "Taller2.lisp")
;; Loading file Taller2.lisp ...
;; Loaded file Taller2.lisp
T
[2]> (resolverforward kb query)
T
```

Figura 7. Pruebas sobre la Knowledge base 1 forward.

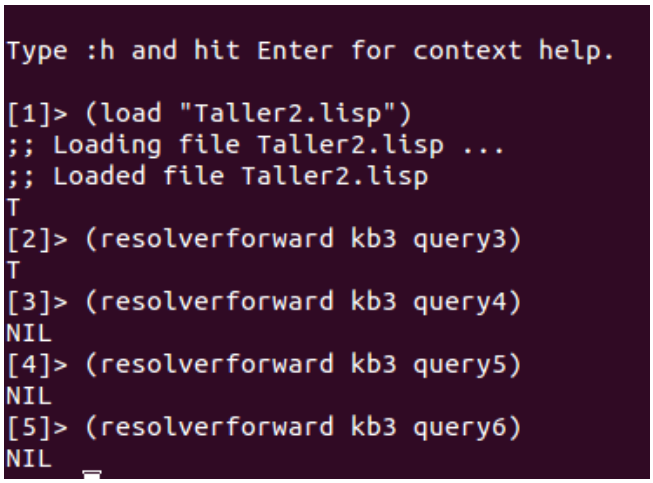

Figura 8. Pruebas sobre la Knowledge base 2 forward.

En esta prueba cuando se utiliza el *query4*, a pesar de que la base de conocimiento tiene información suficiente para responder con verdadero, la estructura recursiva diseñada no logra dar respuesta porque nunca llega a probar ese antecedente y por lo tanto se considera como una restricción del sistema.

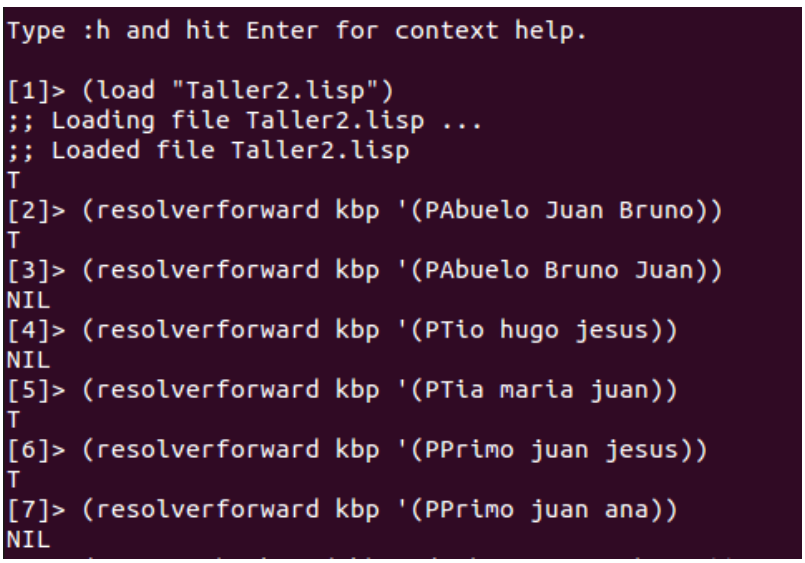

Figura 9. Pruebas sobre la Knowledge base 3 forward.

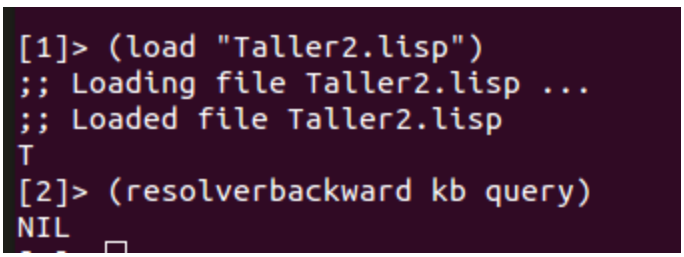

Figura 10. Pruebas sobre la Knowledge base 1 backward.

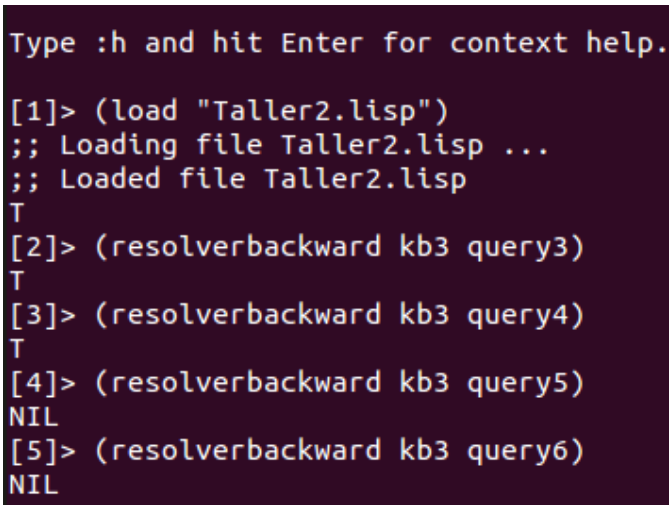

Figura 11. Pruebas sobre la Knowledge base 2 backward.

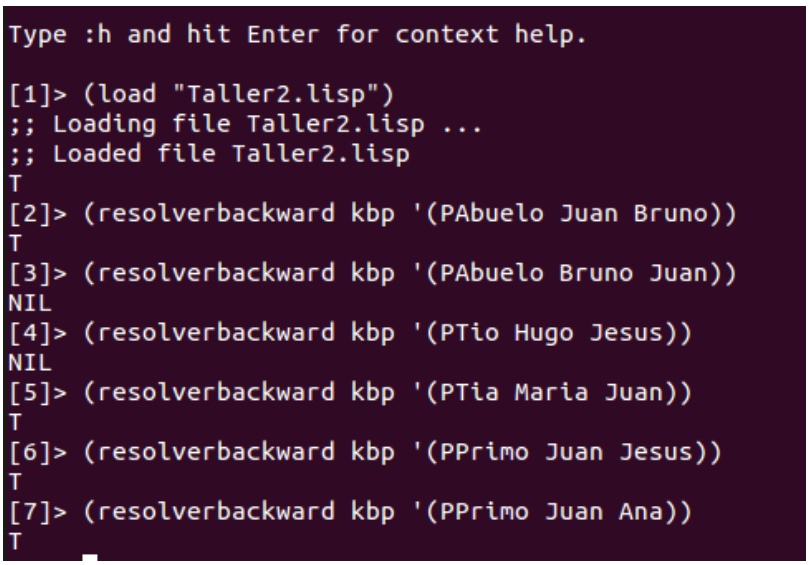

Figura 12. Pruebas sobre la Knowledge base 3 backward.

Conclusiones

- La creación de sistemas que puedan razonar de forma simple es una muy buena práctica para lograr entender la complejidad de estas tareas y lograr reconocer la utilidad de seguir los métodos de razonamiento como forward y backward chaining para problemas de decisión utilizando lógica de primer orden.
- El lenguaje de programación funcional Lisp resulta de mucha utilidad al momento modelar sistemas en Inteligencia Artificial, pues es una herramienta de prototipado muy buena que se adecua fácilmente a problemas en esta rama de las ciencias de la computación, soporta programación simbólica, es extensible, entre otras.
- Realizar este tipo de ejercicios requiere de un buen entendimiento de los algoritmos y de un manejo considerablemente alto de programación funcional, por lo tanto es imperativo dedicarle mucho tiempo para que funcione correctamente.

- Los métodos utilizados para hacer inferencias sobre la base de conocimientos creada en este proyecto son muy básicos e ineficientes, por ende es de esperarse que se realicen procedimientos ineficientes dependiendo de la base de conocimiento para lograr inferir o deducir un *query*.
- Lograr llegar a una solución correcta depende en gran medida de la base de conocimientos, si esta no tiene suficiente información para lograr llegar al resultado, no se lograra llegar y en ciertos casos se podrían dar ciclos infinitos que resultan en una falla.

Bibliografía

1. Apuntes de la clase de Inteligencia Artificial
2. Russell Stuart, Norving Peter. Artificial Intelligence, A Modern Approach. Ed 3. Prentice Hall. 2010. [2] Poole L.