Proyecto 1: Implementación de Tableaux

Integrantes:
Ricardo Rubén González García (315313446)
Correa García José Ángel (315333514)

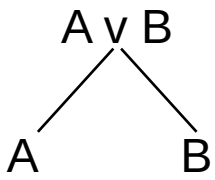
Un método para poder verificar si una formula es una tautología, contradicción, que sea satisfacible o insatisfacible encontrando un estado para el dicha formula sea verdadera o no, es por medio de tablas semánticas o tableaux.

Un tableau consiste en crear un árbol, buscando una interpretación para la formula, mediante algunas reglas iniciales.
Dentro de las reglas iniciales para poder usar el método de Tableau, la formula que queremos analizar no debe contener implicaciones, dobles implicaciones y las negaciones deben de afectar a variables atómicas (¬P,¬(True),¬(False)), por lo que deberemos ocupar equivalencias lógicas y leyes de DeMorgan para pasar de una forma a otra, obteniendo una formula solo con conjunciones, disyunciones y variables atómicas.

Debemos cuidar que los paréntesis la formula no llegue a cambiar el significado de la formula original al transformarla.

Para construir un tableau debemos seguir las siguientes reglas.

1. La formula para que deseamos construir el tableau aparece como raíz del árbol.
2. Si el conector principal de la formula es una disyunción, (A v B), de la raíz del subárbol se abren dos ramas, una pasando la primera formula (A) y para la otra rama la segunda (B)



3. Si la formula en el subárbol en el que nos encontremos es una conjunción (A ^ B), se coloca un operador como hijo del otro.

Esto cubriría las reglas principales para el desarrollo de un tableau, con esto podemos generar un árbol, lo siguiente es darle una utilidad al árbol creado.

Cerrando Ramas

Cada rama que se creo desde la raíz del tableau hasta la punta de ella, representa un posible estado para nuestra formula.

Como dijimos anteriormente, nuestras formulas solo pueden constar de conjunciones, disyunciones y formulas atomicas, ya sean negadas o simples, lo que nos puede generar que en el camino de la punta de una rama a su raíz tengamos variables de la forma ***P*** *y* ***¬P*** por lo que ese posible estado de nuestro tableau genera una contradicción, pues se traduciría a (**P^¬P**) , por lo que todo lo que hay debajo de esta rama no podría satisfacer la formula.
Entonces, cada que encontremos una variable atomica y su negada en la misma rama podemos dejar ese camino y no seguir desarrollando, a esto le llamaremos **Cerrar la rama(!!).**

Esto nos servira al momento de desarrollar los pasos anteriores, que si encontramos una variable con su complemento en un punto, lo cerremos y dejemos de seguir desarrollando por ese camino, lo que nos ahorrara trabajo al crear el tableau.

Interpretación de ramas cerradas

Si todas las ramas quedan cerradas, tenemos que la formula original es una contradicción, es decir que para todos los posibles estados, no hay alguno que pueda satisfacer a la formula.
Que alguna rama quede abierta, es que podemos dar un estado que satisfaga a la formula original usando interpretaciones.
Por otro lado, si todas las ramas quedan abiertas, no significa que sea una tautología.
Para comprobar esto, debemos crear un nuevo tableau negando la formula original.
Si en este segundo caso, todas las ramas se cierran, tenemos que la negación de la formula original es una contradicción, lo que significa que la formula original es una tautología.

Reglas α y β

Estas reglas nos permitirán ir desarrollando la formula conforme la incluyamos al tableau, lo que hará que si encontramos en algún punto la variable con su complemento, nos detengamos ahí, y no sea necesario pasar toda la formula a su forma que mencionamos anteriormente. Las reglas son las siguientes:

α-Reglas

1. De A ^ B se deduce A y B**.**----------α(1)
2. De ¬(A v B) se deduce ¬ A y ¬ B.----α(2)
3. De ¬ (A v B) se deduce A y ¬ B.------α(3)

β-Reglas

1. De A v B se deduce A y, en una rama separada, B. ----------β(1)
2. De ¬ (A ^ B) se deduce ¬ A y, en una rama separada, ¬ B.----β(2)
3. De A → B se deduce ¬A y, en otra rama separada, B.----------β(3)

Por ultimo es la regla de cierre que hace alución al cerrar ramas.

1. Cerrar cualquier rama que tenga A y ¬ A (Para cualquier A), o bien tenga ¬ True o False.

Desarrollo de la implementación.

El primer problema que enfrentamos fue como pasar una proposición a un tipo de dato sobre el que podamos ir haciendo recorridos. El primer intento fue hacerlo con listas pues pensamos que seria mas fácil, pero llegamos a un punto en que no sabíamos como tratar con las disyunciones, pues algo de un lado de una disyunción nos afectaba el otro lado. Decidimos cambiarnos a arboles y fue entonces que solucionamos que un lado de la disyunción afectara al otro.

Después de eso nos enfrentamos a otro problema, ya teníamos toda la proposición traducida a arboles, ya teníamos las hojas, ahora solo quedaba cerrar ramas.
Sin embargo aquí fue donde encontramos el siguiente error, pues no teníamos una forma para poder acceder al árbol por encima del nodo que teníamos, el Tree que definimos en el laboratorio fue insuficiente.
Pensamos en hacer un

type Tree a = Empty | Branch a (Tree a) (Tree a) (Tree a)

Siendo el primer Tree que aparece el arbol que tenemos por encima al nodo en el que estamos actualmente; tuvimos problemas a la hora de definir una hoja pues no podíamos poner un Arbol genérico como padre, pero seguimos sobre este esquema, pensando que podríamos ir definiendo cada hoja cuando se necesitara. Después tuvimos mas problemas a la hora de hacer las funciones de alfaRegla y betaRegla asi que decimos dejarlo ahi.
Empezamos a buscar en LearnHaskell y ahi nos topamos con los Zipper y decidimos que esa seria la forma de seguir con el proyecto ya que un Zipper no guarda solamente el árbol actual (y sus subarboles), sino que también guarda información de su árbol "hermano" y de su árbol "padre", poder ir hacia arriba en un árbol ya resultaba posible, por lo que ya podíamos ir cerrando ramas. En este caso la información del árbol hermano la desechamos, pues no nos importaba que pasaba con el.
Ya con los Zippers seguimos trabajando y, aunque un poco engorroso, pudimos seguir haciendo las definiciones e incluso llegamos a terminar un

satisfacible :: Prop → Bool y un tautologia :: Prop → Bool

Sin embargo fallaban en casos muy sencillos y solo daban lo que se esperaba en casos muy específicos y ahí nos dimos cuenta de que algo fallaba.
Despues de revisar por bastante tiempo nos dimos cuenta de que lo que fallaba era el método para cerrar las ramas y era debido a la facilitud con la que nos perdíamos a la hora de trabajar con los Zippers, asi que optamos por hacer un truco.
A cada hoja en su forma Zipper le sacamos una lista desde la hoja hasta la raíz del árbol y para cerrar las ramas tan solo había que ver si había alguna contradicción en el camino.
Al final, para obtener si es satisfacible solo comprobamos si alguna de las listas se quedo "abierta" y para saber si es tautología, solo negamos la proposición de entrada y vemos que no queden listas "cerradas"

Demostraciones.

1. **Teorema 1. (Correctud):** Si hay un tableau cerrado para Γ, entonces Γ no tiene un modelo.
2. **Teorema 2. (Completud):** Si Γ no tiene un modelo, entonces existe un tableau T(Γ) cerrado.

1.-Demostración:
Haremos la demostración por contradicción, por lo que suponemos que Γ tiene un modelo.
Si Γ tiene un modelo, significa que hay una interpretación para la formula dada, lo que implica que alguna de las ramas te su tableu asociado esta abierta, pero esto es una contradicción, pues empezamos suponiendo que el tableu de Γ es cerrado.
Por o tanto, se cumple que si hay un tableo cerrado para Γ, entonces Γ no tiene un modelo.

2.-Demostración:
Haremos la demostración por contrapositiva. Lo que deberemos demostrar es que si existe un tableu de Γ abierto, entonces Γ tiene un modelo.
Como T(Γ) es abierto, podemos seguir el camino de una de las ramas para llegar a su hoja abierta y asignando valores a nuestras variables de tal forma que genere nuestra interpretación. Por lo tanto, todo tableu abierto tiene un modelo.