Prueba Práctica de Paradigmas de Programación

"Tablas Semánticas"

Introducción

Para representar expresiones de la lógica proposicional con variables, se han diseñado los siguientes tipos de datos:

```
type oper = Not;;

type biOper = Or | And | If | Iff;;

type prop =
    C of bool
  | V of string
  | Op of oper * prop
  | BiOp of biOper * prop * prop;;
```

De este modo, por ejemplo, la proposición $(p \to q) \Leftrightarrow (\neg p \vee q)$ estaría representada por el valor:

```
BiOp (Iff, BiOp (If, V "p", V "q"), BiOp (Or, Op (Not, V "p"), V "q"))
```

Para calcular el valor de cualquiera de estas proposiciones en un determinado contexto o lista de pares `(<variable>, <valor-booleano>)`, se han implementado las siguientes funciones:

```
let opval = function
    Not -> not;;

let biopval = function
    Or -> (||)
  | And -> (&&)
  | If -> fun p q -> (not p) || q
  | Iff -> (=);;

let rec peval ctx = function
    C b -> b
  | V s -> List.assoc s ctx
  | Op (op, p) -> (opval op) (peval ctx p)
  | BiOp (biop, p1, p2) -> (biopval biop) (peval ctx p1) (peval ctx p2);;
```

De esta forma, una función que indique si una proposición lógica es o no una tautología podría implementarse como sigue:

```
let rec vars = function
    C _ -> []
  | V s -> [s]
  | Op (_, p) -> vars p
  | BiOp (_, p1, p2) -> vars p1 @ vars p2;;

let rec remove_dups = function
    [] -> []
  | h::t ->
```

```
      if List.mem h t then remove_dups t
      else h :: (remove_dups t);;

let pvars p =
  remove_dups (vars p);;

let rec ctxs = function
    [] -> [[]]
  | h::t ->
      let cs = ctxs t in
      (List.map (function c -> (h,true)::c) cs) @
      (List.map (function c -> (h,false)::c) cs);;

let is_tau p =
  let cs = ctxs (pvars p) in
  List.for_all (function c -> peval c p) cs;;
```

Podríamos decir que esta implementación resuelve la problemática por "fuerza bruta", ya que extrae las variables de la proposición, genera todos los posibles contextos resultantes de dar valores verdadero o falso a cada variable (es decir, construye las *tablas de verdad*), y evalúa la proposición en todos y cada uno de esos contextos. Si el resultado de todas esas evaluaciones es siempre verdadero, la proposición es una tautología. En caso contrario, no lo es, y cualquiera de los contextos en los que la evaluación da falso constituye un contraejemplo de la proposición.

Sin embargo, para saber si una proposición lógica es o no una tautología, existe un método alternativo que en la mayoría de las ocasiones resulta ser más eficiente. Dicho método utiliza las *tablas semánticas* (en lugar de las tablas de verdad), las cuales se describen a continuación.

Tablas semánticas

En general, el método de las tablas semánticas o árboles semánticos utiliza el principio de *demostración por contradicción* o *reducción al absurdo* para demostrar si un argumento es o no válido. En particular, este método (descubierto de manera independiente por Beth y Hintikka en los años cincuenta) permite saber si una proposición lógica es o no una contradicción. Por tanto, para saber si una proposición lógica es o no una tautología, podemos aplicar este método a la negación de la proposición.

El método crea un árbol cuya raíz es la proposición considerada, y la va descomponiendo en proposiciones más sencillas, aplicando sucesivamente las reglas de la figura 1. Cada proposición descompuesta se marca como ya utilizada, y se sigue trabajando con el resto de nodos del árbol que contengan proposiciones pendientes de tratar. Si en una rama del árbol aparece en algún momento una proposición y su negación, se dice que esa rama queda cerrada. Si al final del proceso todas las ramas se cierran, la proposición es una contradicción. En caso contrario, cada rama abierta constituye un contraejemplo de la proposición inicial.

Las constantes booleanas merecen un tratamiento especial: si en una rama del árbol aparece en algún momento un nodo cuyo valor es $\mathcal{F}$ (*false*), dicha rama puede ser automáticamente cerrada; mientras que si el valor del nodo es $\mathcal{T}$ (*true*), dicho nodo puede ser ignorado y eliminado, y el proceso continúa normalmente con el estudio del resto de nodos.

Las figuras 2 y 3 muestran, respectivamente, la demostración del *modus tollens* y la de una de las *leyes de De Morgan*. La figura 4, por su parte, muestra la refutación del argumento $(p \vee q) \rightarrow p$.

Por último, la figura 5 muestra una demostración optimizada respecto a la de la figura 3. Dentro de una misma rama, los nodos que contienen proposiciones pendientes pueden ser procesados en cualquier orden. Sin embargo, tal y como se observa al comparar estas dos figuras, si damos prioridad a las reglas que retrasan la apertura de ramas, se producen árboles semánticos con la misma profundidad y con el mismo número de ramas, pero con menor número de nodos a tratar.

En realidad, tal y como esbozábamos anteriormente, el propósito del método de las tablas semánticas es más amplio. Así, si queremos demostrar o refutar un argumento del tipo $H \rightarrow C$, calculamos la tabla semántica de $H \wedge \neg C$. Si al finalizar el proceso, todas las ramas se cierran, tenemos que $H \wedge \neg C$ es una contradicción, es decir, el argumento $H \rightarrow C$ es válido. Por el contrario, la existencia de una rama abierta nos llevará a concluir que el argumento no es válido. Del mismo modo, si tenemos un sistema de proposiciones $\{p_1, p_2, \ldots, p_n\}$, sabremos que es consistente, si al construir la tabla semántica de $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ nos queda algún camino abierto, el cual representará un modelo para dicho sistema.

En resumen, las principales ventajas del método de las tablas semánticas respecto al de las tablas de verdad serían las siguientes: es menos costoso de aplicar; constituye una buena base para programar demostradores automáticos; puede extenderse a otras lógicas más potentes que la lógica de proposiciones, para las cuales el método de las tablas de verdad deja de tener sentido; y además, en el caso de que el argumento no sea válido, las tablas semánticas siguen mostrando explícitamente los contraejemplos.

Se puede obtener más información sobre las tablas semánticas en los siguientes enlaces:

- `https://es.wikipedia.org/wiki/Árbol_semántico`
- `https://en.wikipedia.org/wiki/Method_of_analytic_tableaux`

Objetivo e instrucciones de entrega

- En el fichero `logic.ml` proporcionado con este mismo enunciado, reimplemente la función `is_tau` con nombre `is_tau_2`, de forma que utilice el método de las tablas semánticas, y no el de las tablas de verdad.
- Después de inspeccionar el código de las propuestas recibidas, éstas serán calificadas con una nota máxima de 1 punto (a contar dentro de la sección "Pruebas Prácticas" de la evaluación de la asignatura). Este ejercicio puede realizarse individualmente o en grupos de hasta 2 personas. Cuando uno de los ejercicios valorados haya sido realizado por un grupo de 2 personas, a cada miembro del grupo se le asignará un 60% de la nota del ejercicio. La primera línea del fichero a entregar (`logic.ml`) debe contener, como comentario, el nombre completo de todos sus autores.
- La entrega se realizará en el gitlab de la facultad (`https://git.fic.udc.es`). Dentro de él, se creará un nuevo proyecto con nombre `Logic`. Todos los profesores de la asignatura deben ser añadidos con rol Master, al igual que todos los miembros del grupo, en el caso de que el ejercicio no sea resuelto de manera individual. El proyecto debe contener únicamente un archivo de nombre `logic.ml`. Una vez terminado, debe enviarse aviso de esto mediante correo electrónico a las direcciones `jorge.grana@udc.es` y `jose.molinelli@udc.es`. La entrega debe realizarse antes de las 24:00 horas del día 11 de enero de 2021.

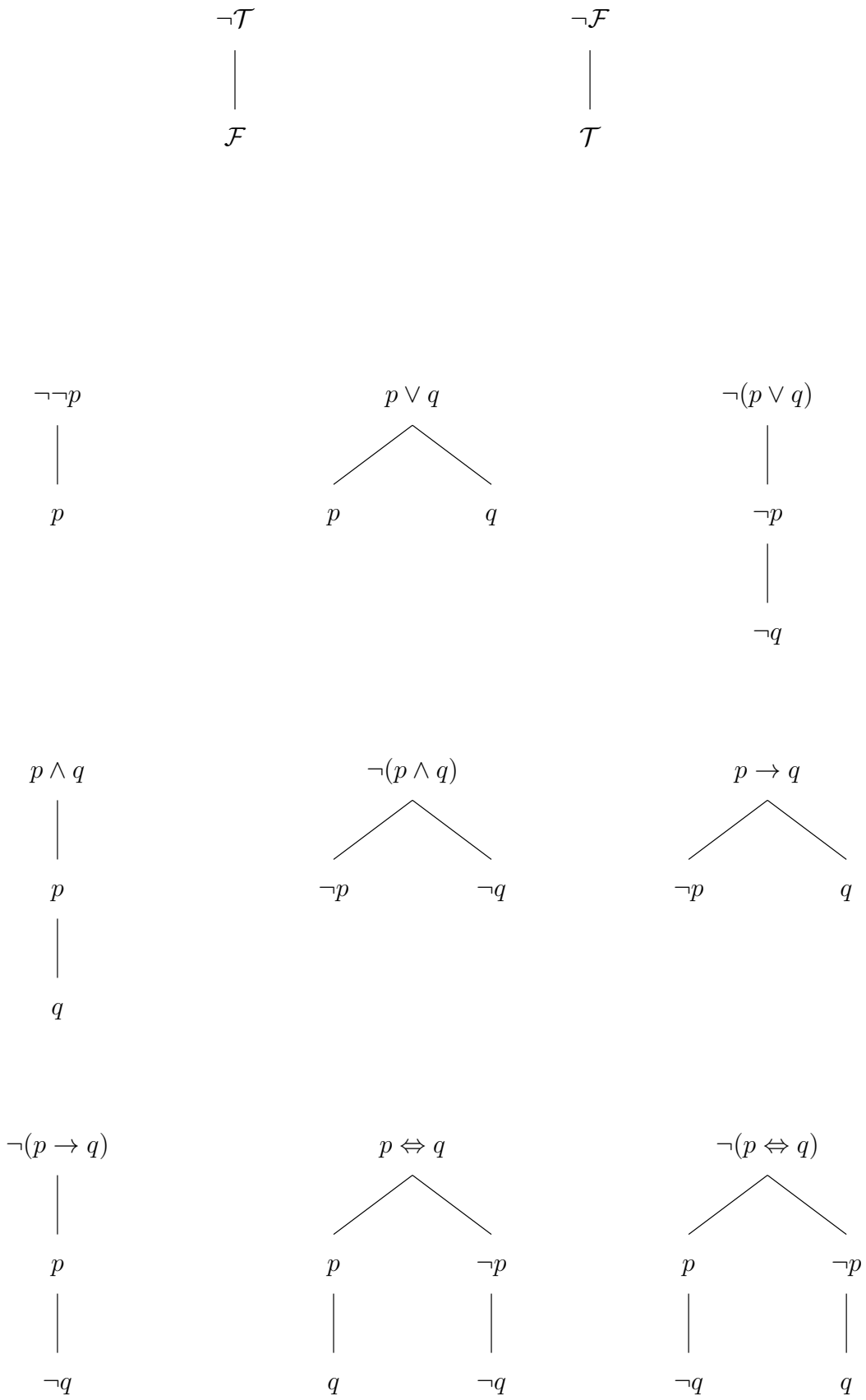

Figura 1: Reglas de descomposición para la construcción de tablas semánticas.

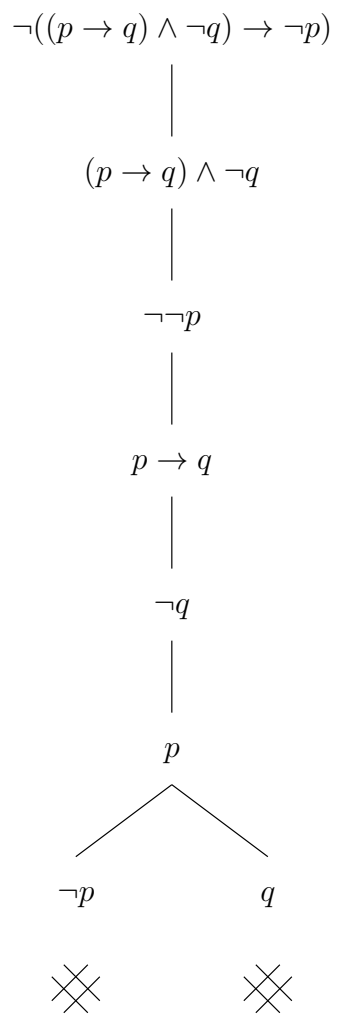

Figura 2: Demostración del *modus tollens*: $((p \to q) \wedge \neg q) \to \neg p$.

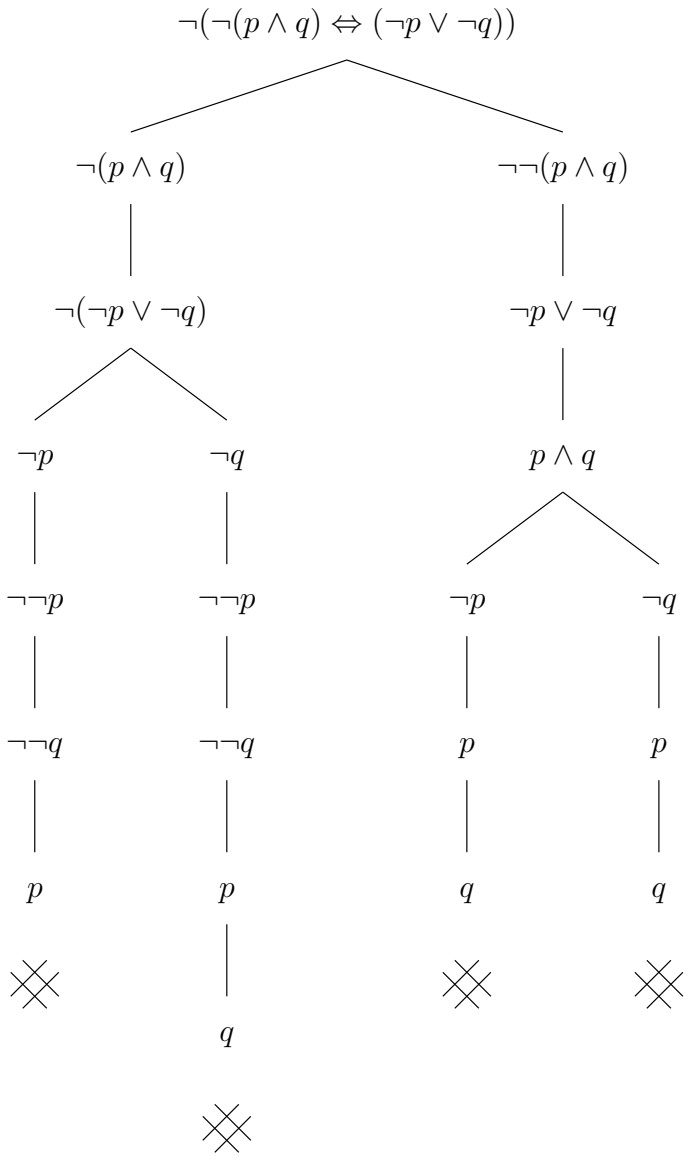

Figura 3: Demostración de una de las *leyes de De Morgan*: $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$.

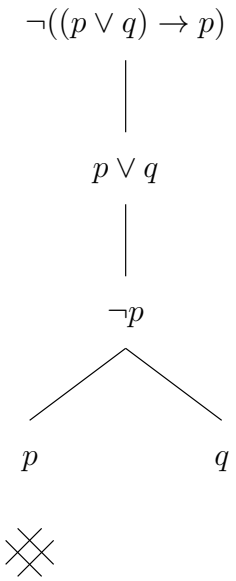

Figura 4: Refutación del argumento $(p \vee q) \rightarrow p$. La rama abierta muestra que $\{\neg p, q\}$ es un contraejemplo, ya que si p es falso y q es verdadero, el argumento es falso.

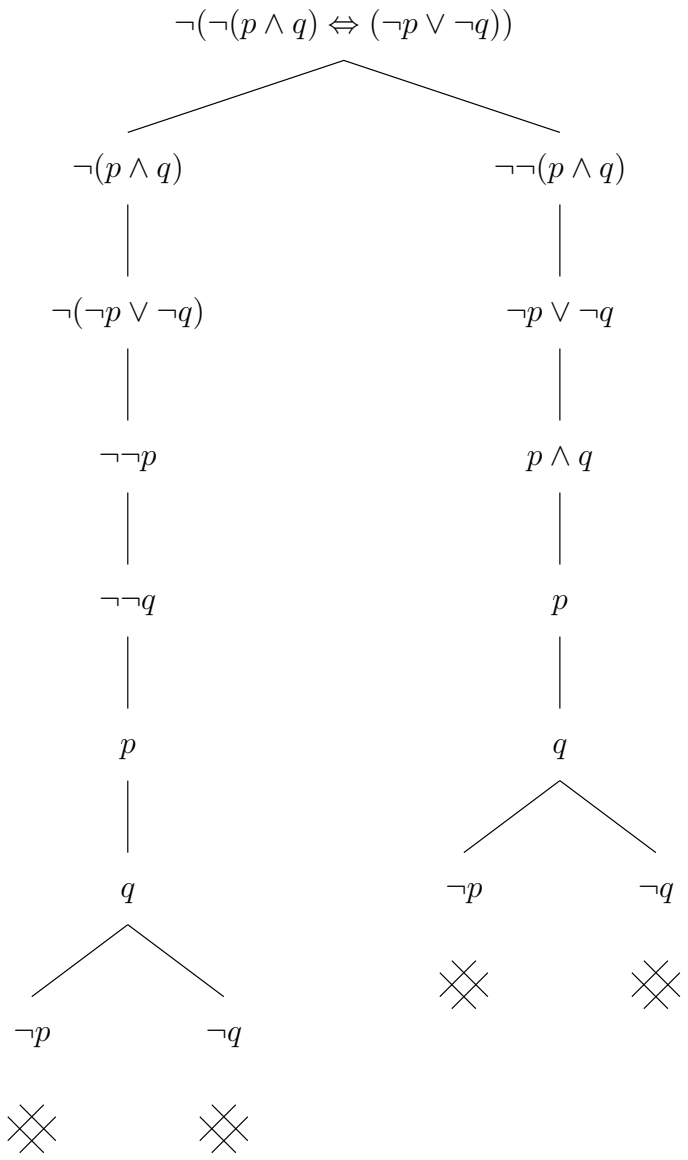

Figura 5: Demostración optimizada de la *ley de De Morgan* de la figura 3.