Argumentos didácticos a favor de los Tableaux Semánticos

Guillermo R. Simari Alejandro G. Stankevicius*

Departamento de Ciencias de la Computación
Universidad Nacional del Sur
Bahía Blanca - Buenos Aires - ARGENTINA
e-mail: {grs,astank}@criba.edu.ar

Resumen

Actualmente existe consenso en la comunidad académica sobre la importancia de incorporar al menos los conceptos básicos acerca de los Sistemas Formales dentro de la currícula de grado en Ciencias de la Computación [1]. En general, se alcanza a cubrir tanto la lógica proposicional como la lógica de primer orden. Al abordar cada teoría debemos definir algún sistema de prueba (empleado a la hora de determinar el conjunto de teoremas sancionados por la misma). El enfoque clásico propone adoptar la resolución básica (ground resolution) o las tablas de verdad, para el cálculo proposicional, y resolución general para el cálculo de predicados.

El objetivo de este trabajo consiste en fundamentar la elección del Tableau Semántico Proposicional (TSP) como una alternativa más apropiada desde un punto de vista didáctico a la hora de introducir un sistema de prueba para el cálculo proposicional. Este nuevo enfoque ha sido aplicado por la cátedra de la asignatura "Lógica para Ciencias de la Computación", con satisfactorios resultados. A manera de síntesis de las virtudes a ser expuestas, se desarrollará una implementación abstracta de TSP, apropiada para acompañar la presentación de la teoría, que reduce la distancia entre los conceptos formales que deben ser introducidos y la experiencia previa de los alumnos.

Palabras Clave: LÓGICA, TABLEAU SEMÁNTICO, CÁLCULO PROPOSICIONAL.

*Becario de Introducción a la Investigación, Secretaría General de Ciencia y Tecnología, UNS.

Argumentos didácticos a favor de los Tableaux Semánticos

1. Introducción

Actualmente existe consenso en la comunidad académica sobre la importancia de incorporar al menos los conceptos básicos acerca de los Sistemas Formales dentro de la currícula de grado en Ciencias de la Computación [1]. Dependiendo del tiempo dedicado a la asignatura que cubra estos contenidos, se alcanzará a desarrollar tanto la lógica proposicional como la lógica de primer orden y tal vez se pueda analizar en profundidad algún subconjunto de esta última (por ejemplo, considerando sólo cláusulas Horn).

Al abordar cada teoría debemos definir algún sistema de prueba o método de cómputo, que será empleado a la hora de determinar sistemáticamente el conjunto de teoremas sancionados por la misma. El enfoque clásico adopta la resolución básica (ground resolution) o las tablas de verdad como método de cómputo para el cálculo proposicional, y resolución general para el cálculo de predicados. De acuerdo a nuestras convicciones, existen alternativas más apropiadas que el enfoque clásico, al menos desde la óptica del docente.

El objetivo de este trabajo consiste en fundamentar la elección del Tableau Semántico Proposicional (TSP) como una alternativa más apropiada desde un punto de vista didáctico a la hora de introducir un sistema de prueba para el cálculo proposicional. Este nuevo enfoque ha sido aplicado por la cátedra de la asignatura "Lógica para Ciencias de la Computación", con satisfactorios resultados.

En la siguiente sección se define el marco teórico necesario para poder introducir la citada variante de tableau. En la sección 3 se compara el TSP con otros sistemas de prueba, enumerando las propiedades observadas. Finalmente, en la sección 4, a manera de síntesis de estas virtudes, se desarrolla una implementación abstracta apropiada para acompañar la presentación teórica de estos conceptos. Debemos notar que, al hacerlo, estamos favoreciendo en los alumnos la transición de lo concreto (su experiencia previa, usualmente implementando), a lo abstracto (en este caso, las teorías formales), técnica en la cual parece radicar la esencia del aprendizaje.

2. Cálculo Proposicional

El TSP constituye un sistema de prueba para la lógica proposicional en general; para cualquier conjunto suficiente de conectivos lógicos se puede definir alguna variante de TSP que se desempeñe como método de cómputo del cálculo definido. Tengamos en cuenta que para poder ejemplificar propiedades sería conveniente disponer de alguna variante concreta de TSP. Como éste es el objetivo de la sección 3, debemos definir la lógica proposional que estamos considerando y especificar su correspondiente variante de TSP.

Las siguientes definiciones sobre cálculo proposicional siguen el espíritu de [2], con las modificaciones necesarias para situarlas en el contexto que nos concierne (Davis sigue el enfoque clásico, utilizando resolución básica como sistema de prueba). Lo mismo sucede con las definiciones sobre tableaux tomadas de [3], también adaptadas, ya que Fitting no orienta su texto al nivel introductorio.

2.1. Sintaxis y semántica

Para describir la sintaxis de una lógica proposicional $\mathcal{L}$ debemos definir el conjunto de símbolos de su alfabeto así como el conjunto de reglas de generación de sus fórmulas bien formadas.

1. El alfabeto de $\mathcal{L}$ está compuesto por los siguientes símbolos:
   - Los símbolos $a_1, a_2, \ldots, a_n, \ldots$, que denominaremos letras proposicionales.
   - Los conecivos lógicos $\neg$, $\vee$ y $\wedge$.
   - Los símbolos auxiliares) y (.
2. La clase de las fórmulas bien formadas (fbfs) consiste de:
   - Fórmulas atómicas: esto es, las letras proposicionales $a_1, a_2, \ldots$
   - Fórmulas compuestas: si P y Q son fbfs, entonces $(\neg P)$, $(P \vee Q)$ y $(P \wedge Q)$ también lo serán.

La interpretación para los conectivos lógicos $\neg$, $\vee$, $\wedge$ es la usual y está fijada: negación, disyunción y conjunción respectivamente.

Definición 2.1. Una *interpretación* para $\mathcal{L}$ es una asignación de valores de verdad (falso o verdadero) a cada letra proposicional. ■

Fijando como interpretamos los conectivos lógicos y asociando un valor de verdad a cada letra proposicional estamos definiendo como interpretar una fórmula en general. Recordemos las importantes nociones de *validez* y *demostrabilidad* (las definiciones a continuación corresponden sólo a la teoría formal $\mathcal{L}$).

Definición 2.2. Una interpretación I *modela* a una fbf X sí, y sólo sí, la interpretación I hace verdadera a X. ■

Definición 2.3. Una fbf X es *válida* (notamos $\models X$) sí, y sólo sí, toda interpretación I modela a X. ■

Definición 2.4. Decimos que existe una *deducción* de una fbf X, a partir de un conjunto de premisas S (notamos $S \vdash X$) sí, y sólo sí, existe una secuencia finita $X_1, X_2, \ldots, X_n$ de fbfs tal que $X_n = X$ y todo P_i, $1 < i \leq n$, es un axioma, o bien una fórmula de S, o bien es consecuencia directa de P_j y P_k, $j, k < i$, por virtud de la regla de inferencia modus ponens. ■

Definición 2.5. Una fbf X se dice *demostrable* (notamos $\vdash X$), sí, y sólo sí, existe una deducción de X a partir de un conjunto vacío de premisas. ■

2.2. Tableau Semántico Proposicional

El método de cómputo por tableau semántico, introducido por Smullyan en [4], es análogo a la resolución en el sentido de que ambos son sistemas de refutación. El objetivo de todo sistema de prueba completo y sensato, tal como el TSP, consiste en determinar la validez de una fórmula arbitraria (en el sentido de la definición 2.3).

En principio, un tableau es simplemente un árbol binario con fórmulas bien formadas de $\mathcal{L}$ como etiqueta de sus nodos. La definición formal es inductiva, definiendo una clase de tableaux como atómicos y especificando reglas de formación para los tableaux compuestos.

Intuitivamente, para demostrar la validez de X debemos probar que no es posible satisfacer $\neg X$. Esta tarea se lleva a cabo descomponiendo sistemáticamente la fórmula $\neg X$ en subfórmulas, tratando de identificar una contradicción entre sus componentes. Por ejemplo, para demostrar la validez de $X = P \vee \neg P$, comenzamos considerando $\neg X$. Ahora bien, para satisfacer $\neg(P \vee \neg P)$ debemos satisfacer tanto $\neg P$ como $\neg\neg P$. Sea $P' = \neg P$, en este caso debemos satisfacer P' y $\neg P'$. Esto no es posible, por lo tanto la fórmula original $\neg(P \vee \neg P)$ es insatisfacible y por ende $P \vee \neg P$ es válida (como era de esperarse).

Emplearemos un artificio ideado por el mismo Smullyan para estandarizar el tratamiento de los distintos conectivos lógicos, denominado *notación uniforme*. Debemos diferenciar las fórmulas cuyo conectivo principal actúa conjuntivamente (denominadas α-fórmulas), de las fórmulas en donde su conectivo principal actúa disyuntivamente (denominadas β-fórmulas). Sean P y Q fórmulas arbitrarias de $\mathcal{L}$, entonces podemos caracterizar a las fórmulas α y β de la siguiente manera:

α	α_1	α_2
$P \wedge Q$	P	Q
$\neg(P \vee Q)$	$\neg P$	$\neg Q$

β	β_1	β_2
$P \vee Q$	P	Q
$\neg(P \wedge Q)$	$\neg P$	$\neg Q$

La intuición presentada se captura en las equivalencias $\alpha \equiv \alpha_1 \wedge \alpha_2$ y $\beta \equiv \beta_1 \vee \beta_2$. Esto es, para satisfacer una fórmula α (que actúa conjuntivamente) debemos satisfacer ambas componentes y para satisfacer una fórmula β (que actúa disyuntivamente) debemos satisfacer al menos una de las componentes.

Para poder demostrar la completitud del sistema debemos tener en cuenta un tercer tipo de fórmula, que no actúa conjuntiva ni disyuntivamente, denominadas σ-fórmulas. De no considerarlas, la fórmula $\neg\neg(P \vee \neg P)$, por ejemplo, no podría ser analizada ya que no es posible descomponerla. La equivalencia asociada a este tipo de fórmula es la siguiente:

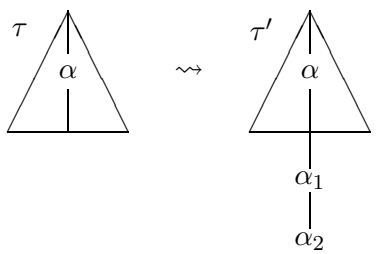
σ	σ_1
$\neg\neg P$	P

Definición 2.6. Denominaremos *tableau semántico* a todo árbol binario, etiquetado con fórmulas de $\mathcal{L}$, que satisfaga la siguiente caracterización inductiva:

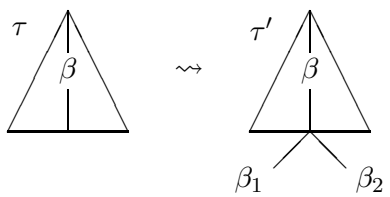
- Si X es una fórmula de $\mathcal{L}$, el árbol compuesto sólo por el nodo raiz etiquetado X es un tableau semántico (denominado tableau para X).
- Si τ es un tableau semántico y τ' es el resultado de aplicar alguna regla de expansión sobre τ, entonces τ' también es un tableau semántico. ■

Definición 2.7. Sea τ un tableau semántico y ρ una rama de τ. Considere las siguientes relaciones entre tableaux:

- Si ρ contiene algún nodo etiquetado con una fórmula α, se puede obtener un tableau τ' a partir de τ donde τ' es similar a τ, salvo que la rama ρ está extendida con dos nodos etiquetados con las fórmulas α_1 y α_2. Gráficamente:



- Si ρ contiene algún nodo etiquetado con una fórmula β, se puede obtener un tableau τ' a partir de τ donde τ' es similar a τ, salvo que la rama ρ bifurca en dos ramas, una con el nodo etiquetado β_1 y la otra con el nodo etiquetado β_2. Gráficamente:



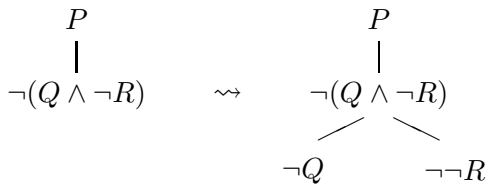
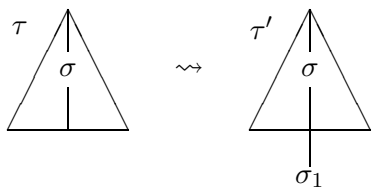


Figura 1: Tableau expansión sobre $\neg(Q \wedge \neg R)$.

- Si ρ contiene algún nodo etiquetado con una fórmula σ, se puede obtener un tableau τ' a partir de τ donde τ' es similar a τ, salvo que la rama ρ está extendida con un nodo etiquetado con la fórmula σ_1. Gráficamente:



Estas relaciones especifican las denominadas *reglas de expansión.* ■

En la figura 1 se aplica una regla de expansión sobre la fórmula $\neg(Q \wedge \neg R)$ del tableau de la izquierda para obtener el tableau de la derecha. Las siguientes definiciones auxiliares proveen la terminología necesaria para describir como se realizan las pruebas por TSP.

Definición 2.8. Sea τ un tableau, ρ una rama de τ y n un nodo de ρ.

- El nodo n está *reducido* en ρ si se verifica alguna de estas situaciones:
  - n está etiquetado con una fórmula atómica.
  - n está etiquetado con una fórmula α y las subfórmulas α_1 y α_2 pertenecen a ρ.
  - n está etiquetado con una fórmula β y al menos una de las subfórmulas β_1 o β_2 pertenece a ρ.
  - n está etiquetado con una fórmula σ y la subfórmula σ_1 pertenece a ρ.
- La rama ρ está *cerrada* si $X \in \rho$ y $\neg X \in \rho$, para alguna fórmula X.
- El tableau τ está *cerrado* si todas sus ramas están cerradas.
- La rama ρ está *terminada* si está cerrada, o bien si todos sus nodos están reducidos.
- El tableau τ está *terminado* si todas sus ramas están terminadas. ■

Contamos con el marco necesario para especificar la forma de construir pruebas por tableau. Esta importante noción se formaliza en las siguientes definiciones.

Definición 2.9. Una *refutación por tableau* para una fórmula X es una secuencia de tableaux $\tau_1, \tau_2, \ldots, \tau_n$ tal que τ_1 es un tableau para X, τ_n es un tableau cerrado y todo tableau τ_i, $1 < i \leq n$, es el resultado de aplicar alguna regla de expansión a τ_{i-1}. ■

Definición 2.10. Una *prueba por tableau* de una fórmula X (notamos $\vdash_{pt} X$) es una refutación por tableau para $\neg X$. ■

$$\begin{array}{c} \neg(\neg A \vee (\neg B \vee A)) \\ | \\ \neg\neg A \\ | \\ \neg(\neg B \vee A) \\ | \\ \neg\neg B \\ | \\ \neg A \\ \otimes \end{array}$$

Figura 2: Prueba por tableau para $\neg A \vee (\neg B \vee A)$.

Desde el punto de vista del docente, pensando en clarificar los ejemplos de pruebas por tableau, conviene adoptar la convención de mostrar sólo el tableau (terminado) resultante. En este sentido, resulta cómodo pensar que las reglas de expansión *transforman* tableaux, en vez de considerar que los generan. Por ejemplo, la figura 2 ilustra una prueba por tableau (de acuerdo a esta convención) para la fórmula $A \to (B \to A)$, expresada sobre los conectivos $\{\neg, \vee, \wedge\}$. Notaremos las ramas cerradas con el símbolo $\otimes$.

El sistema definido no captura naturalmente la noción de prueba en base a premisas (*i.e.*, $S \vdash_{pt} X$). No será necesario extender el sistema en esta presentación, pues lo definido resulta suficiente para discutir las propiedades del TSP; de todas formas, esta situación puede resolverse de manera sencilla incorporando un nuevo tipo de regla: la S-introducción. Por esta regla, en todo momento de la construcción de una prueba por tableau se puede introducir un nuevo nodo a cualquier rama abierta de un tableau y etiquetarlo con alguna fórmula del conjunto S.

Como mencionamos con anterioridad, esta variante de TSP satisface las importantes propiedades de *sensatez* y *completitud*, enunciadas en los siguientes teoremas:

Teorema 2.1. **[sensatez]** *Si $\vdash_{pt} X$, entonces $\models X$. Esto es, si existe una prueba por tableau para la fórmula X, entonces X es válida.*

Teorema 2.2. **[completitud]** *Si $\models X$, entonces $\vdash_{pt} X$. Esto es, si la fórmula X es válida, entonces existe una prueba por tableau para X.*

3. Propiedades y virtudes del TSP

Toda comparación en base a propiedades y virtudes depende en gran medida del objetivo de la comparación. Lo que bajo una comparación por eficiencia puede considerarse una virtud, resulta una desventaja bajo una comparación por claridad. Hemos optado por comparar los sistemas de prueba en base a sus características pedagógicas, en función del rol que pretendemos (acompañar la presentación teórica). Según este criterio, dividiremos las virtudes en dos categorías: aspectos primarios y aspectos secundarios. Bajo este enfoque, por ejemplo, las virtudes devengadas de la eficiencia no tendrán un papel preponderante.

En lo que resta de esta sección, compararemos el TSP contra las propuestas clásicas de sistemas de prueba.

3.1. Aspectos primarios

Esta categoría comprende toda propiedad relevante desde la perspectiva del docente.

◦ Intuitivo

Esta virtud del TSP, que deseamos destacar en primer lugar, surge de su sencilla mecánica. En todo momento sabemos que estamos haciendo: estamos analizando las con-

secuencias de satisfacer una fórmula, con el objeto de determinar si esto es posible. Si satisfacer la fórmula implica satisfacer subfórmulas contradictorias, necesariamente la fórmula original es insatisfacible. Por esta razón, el TSP puede considerarse intuitivo. Ciertamente, el mismo argumento se aplica a las tablas de verdad, pues en todo momento estamos comprobando la validez de una fórmula contra cada interpretación posible.

Esta propiedad no se observa en la resolución básica: podemos generar resolventes durante una prueba que no aporten a la refutación en sí, sin notar diferencia alguna con respecto a las resolventes que de hecho intervienen en ella. Esto se debe a que no es claro el significado de cada paso aplicado en la construcción de la prueba.

◦ Constituye un método gráfico

Hoy en día está fuera de discusión la importancia de contar con herramientas de naturaleza gráfica a la hora de acompañar una presentación teórica. El TSP constituye el sistema de prueba gráfico por excelencia. Las tablas de verdad también pueden ser consideradas "gráficas" en algún sentido, pero de forma no tan directa como el TSP.

Nuevamente, la resolución básica no satisface esta propiedad. Si bien es posible presentar una refutación por resolución de forma gráfica, tan sólo estamos ideando una representación poco natural para una secuencia.

◦ Se puede extender al cálculo de predicados

Como mencionamos durante la introducción, la asignatura que cubra el cálculo proposicional suele abordar también el cálculo de predicados. Por esta razón, el sistema de prueba para el cálculo proposicional debe poder extenderse al cálculo de predicados, ya que introducir un sistema de prueba para tener que descartarlo posteriormente no parece razonable. Esta situación deja fuera de juego a las tablas de verdad, que no pueden extenderse debido a la existencia de dominios infinitos que deben ser considerados como interpretaciones posibles (éstos implican infinitas entradas en la hipotética tabla de verdad asociada).

Cabe destacarse que el tableau semántico se extiende con naturalidad: sólo debemos definir nuevas reglas de expansión para capturar el significado de los cuantificadores al preceder una fórmula (con la intención de poder descomponerla). Se suele denominar γ-fórmulas a las fórmulas que actúan universalmente y δ-fórmulas a las que actúan existencialmente. La denifición puede brindarse mediante reglas, análogas a las empleadas en la sección 2. En [5] se desarrolla una extensión del TSP al cálculo de predicados de acuerdo al enfoque sugerido.

Por otra parte, no podemos ignorar el hecho de que la resolución básica satisface este principio, ya que puede extenderse de forma natural en lo que se denomina resolución en general (incorporando la unificación).

◦ El conjunto de conectivos puede modificarse con facilidad

Resulta sumamente sencillo incorporar un nuevo conectivo lógico al TSP. Por ejemplo, si deseáramos extender el sistema presentado para que admita fórmulas con el conectivo lógico $\rightarrow$, sólo debemos extender apropiadamente el conjunto de reglas que describen las fórmulas α y β, a saber:

α	α_1	α_2
$P \wedge Q$	P	Q
$\neg(P \vee Q)$	$\neg P$	$\neg Q$
$\neg(P \rightarrow Q)$	P	$\neg Q$

β	β_1	β_2
$P \vee Q$	P	Q
$\neg(P \wedge Q)$	$\neg P$	$\neg Q$
$P \rightarrow Q$	$\neg P$	Q

Las tablas de verdad también se pueden extender con igual facilidad, simplemente definiendo mediante una tabla la interpretación deseada para el nuevo conectivo.

Por último, como la resolución básica sólo soporta los conectivos lógicos presentes en las fórmulas en notación fnc (forma normal conjuntiva), la posibilidad de contemplar nuevos conectivos se realiza mediante reglas de reescritura que deben aplicarse antes de comenzar la prueba. Por esta razón, consideramos que en este caso el conjunto de conectivos no puede modificarse (al menos con facilidad).

∘ Se aplica a fórmulas en general

Esta propiedad la satisfacen el TSP y las tablas de verdad. Cualquier fórmula bien formada puede ser suministrada a estos sistema de prueba directamente, sin conversiones previas. No sucede lo mismo con la resolución básica, que en general requiere su entrada en forma normal conjuntiva. Existen algunas variantes de resolución básica que admiten una entrada sin restricciones, por ejemplo la definida en [3], pero el formalismo termina siendo pobre en claridad (la conversión a **fnc** se realiza en paralelo a la refutación por resolución).

Si la prueba de una fórmula no se realiza sobre ella (este será el caso si usamos su equivalente en notación fnc), resulta difícil interpretar el significado de las acciones aplicadas en la construcción de la prueba con respecto a la fórmula original. Esto puede llegar a constituir un impedimento en la asimilación de los conceptos (principalmente al abordar por primera vez los sistemas de prueba).

∘ Si no puede probar una fórmula, fundamenta su no validez

En la demostración del teorema 2.2 (completitud del TSP) se suele probar que a partir de una rama terminada pero no cerrada se puede extraer una interpretación que modele a la fórmula que etiqueta el nodo raíz del tableau. Si el tableau está terminado pero no cerrado debe existir alguna rama en tales condiciones. Esto implica que cuando el TSP no puede probar una fórmula X, ya que no existe un tableaux cerrado para $\neg X$, entonces encuentra una interpretación que modele a $\neg X$ y esta interpretación sirve de contraejemplo de la hipótesis "X es válida". Las tablas de verdad satisfacen de igual manera este principio.

Esta propiedad constructivista no es satisfecha por la resolución básica, donde la ausencia de una refutación por resolución para $\neg X$ debe considerarse prueba *ad hoc* de la no validez de X.

3.2. Aspectos secundarios

Esta categoría comprende el resto de las propiedades deseables, algunas muy importantes si modificáramos el criterio de comparación, pero pertenecientes a un segundo plano desde la perspectiva adoptada.

∘ Eficiencia

La eficiencia, gran paragüa detrás del cual se han escondido una infinidad de aberraciones a la claridad, ha quedado relegada al segundo plano en esta discusión. Podemos considerar que la eficiencia juega un papel poco relevante durante una presentación teórica. El alumno está más ocupado en comprender los nuevos conceptos, que en analizar el rendimiento del sistema introducido.

Hecha esta observación, debemos destacar que se suele considerar al TSP más eficiente que las tablas de verdad. Por ejemplo, una fórmula que contenga n letras proposicionales distintas necesita 2^n entradas para su tabla de verdad, pero una prueba por tableau de la misma fórmula puede requerir un número menor de expansiones. La postura opuesta

se fundamenta en [6], caracterizando una clase de fórmulas donde las tablas de verdad se desempeñan mejor que el TSP.

La conclusión es que estos sistemas no se pueden comparar entre sí. Existen clases de fórmula donde un sistema es mejor que el otro y viceversa. Esto se debe a que la complejidad de las pruebas por tableau depende de la estructura de la fórmula y en las tablas de verdad depende de la cantidad de letras proposicionales. Desde el punto de vista que adoptamos en esta comparación no podemos diferenciarlos (la clase de fórmulas empleadas como ejemplos son de estructura simple y con poca cantidad de letras proposicionales).

Con respecto a la resolución básica, los análisis de eficiencia son arduos, y no existe una clara relación entre algún aspecto de la fórmula considerada y la complejidad de la prueba.

∘ Se puede extender a lógicas modales y lógicas multivaluadas

Smullyan introdujo dos tipos de tableau: los tableaux cuyos nodos están etiquetados simplemente con fórmulas (el TSP definido en la sección 2.2) y los que etiquetan nodos con *fórmulas signadas*. Una fórmula signada es una fórmula bien formada precedida de un signo T o de un signo F, que denotan el valor de verdad pretendido para la fórmula. Las reglas de expansión se adaptan para manejar fórmulas signadas; la fórmula, por ejemplo, $F(P \vee Q)$ se puede expandir en $F(P)$ y $F(Q)$. La ramas se cierran ahora ante la presencia de dos nodos etiquetados $T(X)$ y $F(X)$, para alguna fórmula X.

Este tipo de tableau se puede extender naturalmente a las lógicas multivaluadas o a las lógicas modales admitiendo otros signos además de T y F. Por ejemplo, en [7] se desarrolla un sistema de prueba basado en tableau para lógicas finitamente valuadas y en [8] se analiza una extensión de tableau semántico a la lógica modal.

∘ Se puede adaptar a lógicas no monótonas

Recientes trabajos [9, 10] han propuesto una extensión de los tableaux semánticos como sistema de prueba para una teoría no monótona en particular, la circumscripción. Si bien la extensión se aplica al tableau para la lógica de primer orden, la propiedad que permitió esta extensión, su naturaleza constructivista, es compartida por todas las variantes.

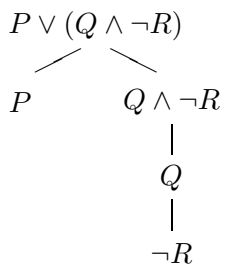
Para determinar la validez de una fórmula circumscripta sólo necesitamos considerar los modelos mínimos de una fórmula, ignorando el resto (*i.e.*, la circumscripción de una fórmula puede ser falsa en un modelo no mínimo y continuar siendo válida). La idea es admitir como prueba de una fórmula X tanto los tableaux cerrados como algunos tableaux abiertos. Recordemos que un tableau abierto para X genera un modelo de $\neg X$. Los tableaux abiertos que se admiten son aquellos cuyos modelos de $\neg X$ no son minimales.

4. Un ejemplo de aplicación

Presentar una implementación abstracta del TSP puede disipar las dudas que persistan sobre sus cualidades, si es que resta alguna luego de la discusión de la sección anterior. El principal atractivo de esta implementación es su alta declaratividad, casi copiando una descripción informal del método, con la ventaja de ser ejecutable.

Utilizaremos Prolog como lenguaje de implementación. La elección puede ser fundamentada bajo diversos argumentos; hemos considerado principalmente dos razones:

- Prolog posee un alto grado de abstracción, permitiendo que nos concentremos en modelar el sistema en vez de tener que lidiar con los detalles de implementación. Recordemos que el objetivo no es la implementación en sí, sino que ésta sea un complemento para facilitar la asimilación de los conceptos teóricos.




```
[ [P o (Q y no R), P], [P o (Q y no R), Q y no R, Q, no R] ]
```

Figura 3: Representación de tableaux mediante lista de ramas.

- El paradigma de *programación en lógica* es el más apropiado a aplicar, ya que en esta asignatura será desarrollada su base teórica (*i.e.*, el cálculo de predicados). En consecuencia, se suele seleccionar Prolog como lenguaje en el cual desarrollar sus clases prácticas. Finalmente, por qué no emplearlo en las clases teóricas también.

El sistema implementado es análogo al presentado. Se puede lograr un entendimiento intuitivo sobre la mecánica del programa aún sin explicitar las suposiciones necesarias para asegurar la correctitud de su funcionamiento. Más aún, resulta un atractivo ejercicio dejar por cuenta del alumnado la demostración de estas propiedades.

En primer lugar discutiremos la elección de una representación interna apropiada para los tableaux, luego analizaremos la resolución planteada y posteriormente la implementación propiamente dicha. Finalmente, extenderemos el sistema implementado para permitir un nuevo conectivo lógico (la facilidad de esta tarea ha sido mencionada como una virtud deseable).

4.1. Representación interna

La definición 2.6 establece que un tableau es simplemente un árbol binario. Si bien Prolog provee soporte para estas estructuras de datos (mediante functores), optaremos por representar el tableau a través de la lista de sus ramas. A su vez, cada rama será representada mediante una lista de fórmulas, ya que una rama es una secuencia de nodos, y cada nodo contiene una fórmula. Por ejemplo, en la figura 3 se aprecia un tableau y su correspondiente representación como lista de ramas. La razón que fundamenta esta elección será discutida junto a la resolución propuesta.

4.2. Resolución

Una prueba por tableau para una fórmula `FBF` consiste en una refutación para `no FBF` (de acuerdo a la definición 2.10). Según nuestra representación, el tableau para `no FBF` se denota por la lista de sus ramas. Su única rama consiste en la fórmula `no FBF`.

```
teorema(FBF) :- refutable( [ [ no FBF ] ] ).
```

La pregunta a responder es ¿bajo qué condiciones un tableau es refutable?. Si pensamos una solución recursiva, un tableau es refutable si algún tableau que se pueda derivar a partir de él (mediante las reglas de expansión) es refutable. El caso base resulta trivial, todo tableau cerrado es refutable.

```
refutable(Tbl) :- derivar(Tbl, NuevoTbl), refutable(NuevoTbl).
refutable(Tbl) :- cerrado(Tbl).
```

El problema fue descompuesto en dos partes (caso inductivo y caso base). Cada alternativa introduce un nuevo subproblema, de complejidad menor y con la ventaja de

representar problemas conceptualmente atractivos. Analicemos en primer lugar el caso base; debemos determinar bajo que condiciones un tableau es cerrado. La definición 2.8 especifica que todas sus ramas deben estar cerradas y que una rama está cerrada si contiene al menos dos nodos etiquetados con fórmulas complementarias. Expresando esta definición como cláusulas PROLOG resulta:

```
cerrado( [] ).
cerrado( [Rama | Resto] ) :-
        miembro(FBF, Rama), miembro(no FBF, Rama), cerrado(Resto).
```

El predicado auxiliar `miembro/2` denota la relación de pertenencia. Puede apreciarse una de las razones por las cuales adoptamos esta representación interna: ha permitido una sencilla especificación recursiva para los tableaux cerrados.

Debemos resolver aún el caso general del predicado recursivo `refutable/1`. Esto es, debemos especificar como se pueden derivar nuevos tableaux en base a las reglas de expansión. Consideremos las fórmulas α; si en el tableau `Tbl` existe alguna rama `Rama`, donde a su vez existe una α-fórmula `FBF`, entonces podemos derivar un tableau, similar a `Tbl`, con la rama `Rama` extendida con las subfórmulas `ALFA1` y `ALFA2`.

```
derivar(Tbl, [ [ALFA1, ALFA2 | Rama] | TblSinR ] ) :-
        miembro(Rama, Tbl), miembro(FBF, Rama), noreducida(FBF, Rama),
        alfa(FBF, ALFA1, ALFA2), eliminar(Rama, Tbl, TblSinR).
```

Los predicados que aparecen en el cuerpo de la regla se deben interpretar de la siguiente manera:

- `noreducida/2`: verifica si la fórmula (primer argumento) está reducida en la rama (segundo argumento).
- `alfa/3`: se satisface si la fórmula (primer argumento) actúa conjuntivamente (*i.e.*, es una α-fórmula), en cuyo caso retorna las componentes α_1 y α_2.
- `eliminar/3`: elimina un elemento (primer argumento) de una lista (segundo argumento), retornando la lista resultante. Según nuestra representación interna, interpretamos que elimina una rama de un tableau retornando el tableau resultante.

Se ha impuesto la restricción de que la fórmula seleccionada no esté reducida en la rama para evitar lazos infinitos. Resulta sencillo demostrar que el sistema con esta restricción preserva la completitud. De hecho, cuando se desarrolla un ejemplo de prueba por tableau a mano, no se requiere aplicar más de una vez regla de expansión alguna sobre una misma fórmula.

Los casos restantes del predicado `derivar/2` son análogos luego de definir los predicados `beta/3` y `sigma/2`. En esencia, para las β-fórmulas debemos incorporar dos nuevas ramas y para las σ-fórmulas sólo una.

Del problema original, verificar que fórmulas del cálculo proposicional son probables por tableau, sólo resta determinar cuando una fórmula no está reducida en una rama (`noreducida/2`) y descomponer las fórmulas de acuerdo a las reglas α, β y σ. Según la definición 2.8, las condiciones bajo las cuales una fórmula está reducida en una rama dependen del tipo de la fórmula. Por ejemplo, una fórmula α está reducida en una rama determinada si ambas subfórmulas aparecen en esa rama. Como queremos especificar cuando **no** está reducida, debemos verificar si al menos una de ellas no aparece en la rama. Esta disyunción se expresa mediante dos reglas. El tratamiento para los otros casos es similar.

```
noreducida(FBF, Rama) :-
        alfa(FBF, ALFA1, ALFA2), nomiembro(ALFA1, Rama).
noreducida(FBF, Rama) :-
        alfa(FBF, ALFA1, ALFA2), nomiembro(ALFA2, Rama).
```

Finalmente, para determinar el tipo y las componentes de una fórmula arbitraria, basta con expresar mediante hechos las reglas que definen los distintos tipos de fórmula y dejar que la unificación haga el resto. Por ejemplo, para las fórmulas α las cláusulas correspondientes son:

α	α_1	α_2	
$P \wedge Q$	P	Q	$\Rightarrow$ `alfa(P y Q, P, Q).`
$\neg(P \vee Q)$	$\neg P$	$\neg Q$	$\Rightarrow$ `alfa(no(P y Q), no P, no Q).`

4.3. Implementación

Aclarados los detalles de la solución propuesta, sólo resta definir algunos aspectos menores para obtener una versión ejecutable. Debemos incorporar las definiciones de los operadores (`y`, `o` y `no`) y de los predicados auxiliares.

```
:- op(101,xfy,y).       :- op(101,xfy,o).       :- op(100,fy,no).

teorema(FBF) :- refutable( [ [ no FBF ] ] ).

refutable(Tbl) :- cerrado(Tbl).
refutable(Tbl) :- derivar(Tbl, NuevoTbl), refutable(NuevoTbl).

derivar(Tbl, [ [ALFA1, ALFA2 | Rama ] | TblSinR ] ) :-
        miembro(Rama, Tbl), miembro(FBF, Rama), noreducida(FBF, Rama),
        alfa(FBF, ALFA1, ALFA2), eliminar(Rama, Tbl, TblSinR).
derivar(Tbl, [ [BETA1 | Rama], [ BETA2 | Rama] | TblSinR ] ) :-
        miembro(Rama, Tbl), miembro(FBF, Rama), noreducida(FBF,Rama),
        beta(FBF, BETA1, BETA2), eliminar(Rama, Tbl, TblSinR).
derivar(Tbl, [ [SIGMA1 | Rama ] | TblSinR ] ) :-
        miembro(Rama, Tbl), miembro(FBF, Rama), noreducida(FBF, Rama),
        sigma(FBF, SIGMA1), eliminar(Rama, Tbl, TblSinR).

cerrado( [] ).
cerrado( [Rama | Resto] ) :-
        miembro(FBF, Rama), miembro(no FBF, Rama), cerrado(Resto).

noreducida(FBF, Rama) :-
        alfa(FBF, ALFA1, ALFA2), nomiembro(ALFA1, Rama).
noreducida(FBF, Rama) :-
        alfa(FBF, ALFA1, ALFA2), nomiembro(ALFA2, Rama).
noreducida(FBF, Rama) :-
        beta(FBF, BETA1, BETA2),
        nomiembro(BETA1, Rama), nomiembro(BETA2, Rama).
noreducida(FBF, Rama) :-
        sigma(FBF, SIGMA1), nomiembro(SIGMA1, Rama).
```

```
alfa(P y Q, P, Q).              alfa(no (P o Q), no P, no Q).
beta(P o Q, P, Q).              beta(no (P y Q), no P, no Q).
gamma(no no P, P).

miembro(X, [X|_] ).
miembro(X, [Y|Ys] ) :- miembro(X,Ys).

nomiembro(X, [] ).
nomiembro(X, [Y|Ys] ) :- X\=Y, nomiembro(X,Ys).

eliminar(X, [X|Xs], Xs).
eliminar(X, [Y|Ys], [Y|Zs] ) :- eliminar(X,Ys,Zs).
```

4.4. Extensión a más conectivos lógicos

Una de las atractivas propiedades del TSP resultó ser la facilidad con la cual se extiende el conjunto de conectivos lógicos. Veamos un ejemplo concreto: el sistema implementado soporta los conectivos $\{\neg, \vee, \wedge\}$ (representados por `no`, `o` e `y` respectivamente); para incorporar el conectivo $\rightarrow$ (representado por `->`), sólo debemos contemplar las nuevas posibilidades (ver sección 3.1), además de declarar el nuevo operador.

```
:- op(102,xfy,->).

alfa(no (P -> Q), P, no Q).     beta(P -> Q, no P, Q).
```

5. Conclusiones

El principal aporte de este trabajo consisten en fundamentar las razones por las cuales el método de prueba por tableau semántico constituye una alternativa más apropiada para acompañar la presentación teórica del cálculo proposicional. Analizamos las propiedades y virtudes del método contra las alternativas clásicas en sistemas de prueba proposicionales. En esta discusión identificamos dos categorías de propiedades, trazando la línea divisoria en función de los intereses del docente. La siguiente tabla resume las virtudes de mayor relevancia observadas:

Propiedad o Virtud	**Tablas de Verdad**	**Resolución Básica**	**Tableau Semántico Proposicional**
Intuitivo.	×		×
Método gráfico.	×		×
Se extiende a primer orden.		×	×
Conjunto de conectivos flexible	×		×
Aplicable a fórmulas en general	×		×
Constructivista	×		×

Las tablas de verdad comparten gran cantidad de propiedades con el TSP, pero fallan cuando intentamos extenderlas a la lógica de primer orden. La resolución básica no adolece de esta limitación, pero carece de la claridad conceptual que brindan los tableaux semánticos. Por lo tanto, el TSP resulta la alternativa más apropiada con este fin.

Las primeras experiencias aplicando este criterio en la asignatura "Lógica para Ciencias de la Computación", dictada en la Universidad Nacional del Sur, han sido, en coincidencia con lo expuesto, lo suficientemente satisfactorias como para impulsarnos a compartir estos resultados con el resto de la comunidad académica.

Referencias

[1] G. Simari, y M. Falappa, *"Sistemas Formales: Presentación de sus Propiedades Abstractas dentro del Currículum de Ciencias de la Computación"*, a ser publicado.

[2] R. Davis, *"Truth, Deduction and Computation"*, Computer Science Press, 1989.

[3] M. Fitting, *"First-Order Logic and Automated Theorem Proving"*, segunda edición, Springer-Verlag, 1996.

[4] R. Smullyan, *"First-Order Logic"*, Springer-Verlag, 1968.

[5] A. Nerode, y R. Shore, *"Logic for Applications"*, Springer-Verlag, 1993.

[6] M. D'Agostino, *"Are Tableaux an Improvement on Truth-Tables?"*, Journal of Logic, Language and Information, vol. 1:**3**, 235–252, 1992.

[7] W. Carnielli, *"Systematization of Finite Many-Valued Logics through the Method of Tableaux"*, Journal of Symbolic Logic 52, 473–493, 1987.

[8] M. Fitting, *"First-Order Modal Tableaux"*, Journal of Automated Reasoning 4, 191–213, 1988.

[9] I. Niemelä, *"Implementing Circumscription using a Tableau Method"*, in proceedings, European Conference on Artificial Intelligence, 80–84, 1996.

[10] I. Niemelä, *"A Tableau Calculus for Minimal Model Reasoning"*, in proceedings, Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods, 278–294, 1996.