4 | LA LÓGICA PROPOSICIONAL

Las reglas de derivación que se han introducido en el capítulo precedente, son válidas para cualquier fórmula que podamos formar con el lenguaje de la lógica proposicional. Aunque claro, por ahora lo único que sabemos sobre las posibles fórmulas del lenguaje es que, a partir de los átomos proposicionales podemos usar conectivos lógicos para crear fórmulas lógicas compuestas. Esto ha sido intencional, pues el objetivo del capítulo anterior era entender la mecánica de las reglas de la **deducción natural**. Es importante recordar la validez de estos patrones de razonamiento, antes de abordar la lógica proposicional como un lenguaje formal.

Consideren este caso, una aplicación de la regla de derivación $(\rightarrow e)$:

1. $p \rightarrow q$ *premisa*
2. p *premisa*
3. q $(\rightarrow e)$ 1, 2

su aplicación es válida aún si substituimos $p \vee \neg r$ por p y q con $r \rightarrow p$:

1. $p \vee \neg r \rightarrow (r \rightarrow p)$ *premisa*
2. $p \vee \neg r$ *premisa*
3. $r \rightarrow p$

Esta es la razón por la cual escribimos las reglas de prueba como esquemas de razonamiento, donde los símbolos griegos se usan como **meta variables** que pueden ser substituidas por fórmulas del lenguaje.

Meta variables

Ahora debemos precisar lo que queremos decir por *cualquier* fórmula del lenguaje. Lo primero que necesitamos es una fuente no acotada de **variables proposicionales**: $p, q, r, \ldots$ o $p_1, p_2, p_3, \ldots$. La cuestión del si tal conjunto es infinito no debería preocuparnos. El carácter no acotado de la fuente es una forma de confrontar que si bien, normalmente necesitaremos una gran cantidad finita de proposiciones para describir un programa de computadora, no sabemos de antemano cuantos vamos a necesitar. Lo mismo sucede con los lenguajes naturales, como el español. La cantidad de frases que puedo expresar en español es potencialmente infinita, pero basta escuchar a los presentadores de noticias de la televisión para saber que podemos usar una cantidad potencialmente finita de ellas.

Variables proposicionales

Que las fórmulas de nuestra lógica proposicional deben ser por tanto cadenas de caracteres formadas a partir del **alfabeto**:

Alfabeto

$$\{p, q, r, \ldots\} \cup \{p_1, p_2, p_3, \ldots\} \cup \{\neg, \wedge, \vee, \rightarrow, (,)\}$$

es una observación trivial, que no provee la información que necesitamos. Por ejemplo, $(\neg)$ y $pq \rightarrow$ son cadenas construídas a partir de este alfabeto, aunque no tienen sentido en la lógica proposicional. Necesitamos especificar cuales de esas cadenas son fórmulas bien formadas.

Definición 4.1 (Fórmula bien formada)**.** *Las fórmulas bien formadas (fbf) de la lógica proposicional son aquellas, y solo aquelas, que se obtienen aplicando finitamente las reglas de construcción siguientes:*

1. *Todo átomo proposicional* $p, q, r, \ldots$ *y* $p_1, p_2, p_3, \ldots$ *es una fórmula bien formada.*
2. *Si* ϕ *es una fórmula bien formada, también lo es* $(\neg\phi)$.
3. *Si* ϕ *y* ψ *son fbf, también lo es* $(\phi \wedge \psi)$.
4. *Si* ϕ *y* ψ *son fbf, también lo es* $(\phi \vee \psi)$.
5. *Si* ϕ *y* ψ *son fbf, también lo es* $(\phi \rightarrow \psi)$.

Este es el tipo de reglas que le gusta a las computadoras. Una fórmula es bien formada si es posible deducir tal cosa con las reglas anteriores. De hecho, las definiciones inductivas de las fórmulas bien formadas de un lenguaje son tan comunes que suelen expresarse en un formalismo diseñado *ad hoc* para este propósito, las gramáticas en notación **Backus-Naur** [1] (BNF). La definición anterior se expresa de manera más compacta como una gramática BNF:

Notación BNF

$$\phi \quad ::= \quad p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \tag{4.1}$$

donde p denota cualquier átomo proposicional y cada ocurrencia de ϕ a la derecha del símbolo ::= denota una fórmula bien formada previamente construída.

Ejemplo 4.1. *¿Cómo podemos probar que la cadena* $(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$ *es una fbf? Afortunadamente, la gramática de la definición 4.1 satisface el* ***principio de inversión****: aunque la gramática permite la aplicación de cinco cláusulas, solo una de ellas se ha aplicado a lo último. En nuestro caso la última cláusula aplicada fue la cinco dado que la fórmula es una implicación con* $((\neg p) \wedge q)$ *como antecedente y* $(p \wedge (q \vee (\neg r)))$ *como conclusión. Siguiendo el principio de inversión podemos ver que el antecedente es una conjunción (cláusula tres) entre* $(\neg p)$ *y* q*, donde el primer operando es una negación (cláusula 2) de* p *que es una fbf (cláusula uno); al igual que* q*. De igual forma podemos proceder con el consecuente de la expresión original y demostrar que ésta es una fbf.*

Principio de inversión

Los paréntesis en estas expresiones suelen confundirnos un poco, aunque son necesarios pues reflejan el hecho de que las fórmulas tienen una **estructura de árbol**, como se muestra en la figura 4.1. Observen que los paréntesis se vuelven innecesarios en esa estructura, pero son necesarios sin linearizamos el árbol en un cadena de caracteres.

Estructura de árbol

Probablemente la forma más fácil de saber si una expresión es una fórmula bien formada, sea analizando su árbol sintáctico. En la figura 4.1 podemos observar que la raíz del árbol es una implicación, de manera que la expresión en cuestión es, a su nivel más alto una implicación. Ahora basta probar recursivamente que los sub-árboles izquierdo y derecho son también fórmulas bien formadas. Observen que las fórmulas bien formadas en el árbol o bien tienen como raíz un átomo proposicional; o un operador con el número adecuado de operandos. Esto ilustra el carácter inductivo de la gramática definida para la lógica proposicional.

Pensar en términos de árboles sintácticos ayuda a visualizar algunos conceptos estándar de la lógica proposicional, por ejemplo, el concepto de **sub-fórmula**. Dada la expresión del ejemplo 4.1, sus sub-fórmulas son aquellas que corresponden a los sub-árboles de la figura 4.1. Esto incluye las hojas p y q que ocurren dos veces; así como r. Luego $(\neg p)$ y $((\neg p) \wedge q)$ en el sub-árbol izquierdo de la implicación. Así como $(\neg r)$, $(p \vee (\neg r))$ y $((p \wedge (q \vee (\neg p))))$ en el sub-árbol derecho de la implicación. El árbol entero es un sub-árbol de si mismo. Así que las nueve sub-fórmulas de la expresión son:

Sub-fórmula

- p
- q
- r

1 John Backus nos dio otros regalitos: Fortran, Algol y los principios de la programación funcional expresados en su lenguaje FP.

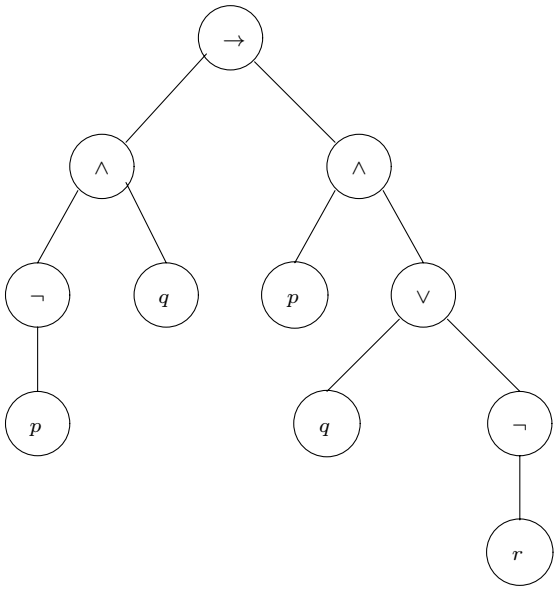

Figura 4.1: El árbol sintáctico de la fórmula bien formada del ejemplo 4.1.

- $(\neg p)$
- $((\neg p) \wedge q)$
- $(\neg r)$
- $(p \vee (\neg r))$
- $((p \wedge (q \vee (\neg p))))$

Ejemplo 4.2. *Consideren el árbol de la figura 4.2 ¿Porqué representa una fórmula bien formada? Todas sus hojas son átomos proposicionales: p dos veces; q y r una. Todos sus nodos internos son operadores lógicos (¬ dos veces, $\wedge$, $\vee$ y $\rightarrow$) y el número de sus sub-árboles es el correcto en todos los casos (un sub-árbol para la negación, dos para los demás operadores). La expresión linearizada del árbol puede obtenerse recorriendo el árbol de manera en orden [2]:* $((\neg(p \vee (q \rightarrow (\neg p)))) \vee r)$.

Ejemplo 4.3. *El árbol de la figura 4.3 no representa una fórmula bien formada. Hay dos razones para ello: primero, las hojas $\wedge$ y ¬, lo cual puede arreglarse diciendo que el árbol está parcialmente construído; segundo, y definitivo, el nodo para el átomo proposicional p no es una hoja, es un nodo interno.*

4.1 SEMÁNTICA

En el capítulo anterior desarrollamos un cálculo del razonamiento que nos permite verificar si las inferencias de la forma $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ son válidas, lo cual quiere decir que a partir de las premisas $\phi_1, \phi_2, \ldots, \phi_n$ podemos concluir ψ. En esta sección abordaremos otra faceta de la relación entre las premisas y la consecuencia de las inferencias, conocida como **consecuencia lógica**. Para contrastar, denotaremos esta relación como:

Consecuencia lógica

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

2 También es posible obtener representación en pre-orden, como las usadas en Lisp, o post-orden; cambiando el recorrido del árbol.

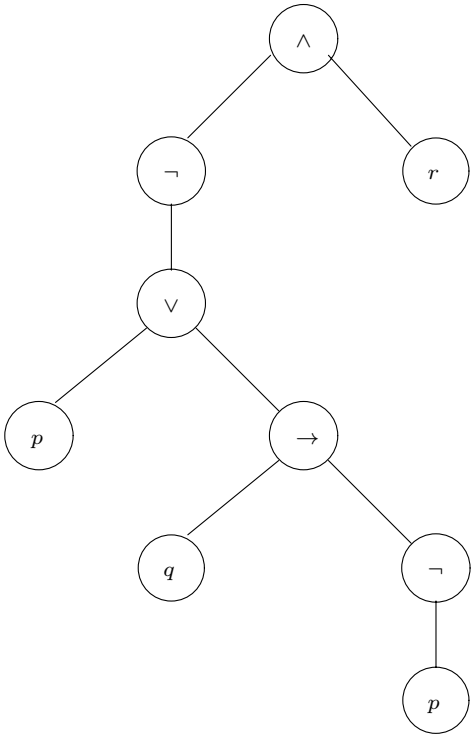

Figura 4.2: Otro ejemplo de un árbol sintáctico de una fórmula bien formada.

La consecuencia lógica se basa en los valores de verdad de las proposiciones atómicas que ocurren en las premisas y la conclusión; así como la forma en que los operadores lógicos manipulan estos valores de verdad. Pero ¿Qué es un valor de verdad? Recuerden que los enunciados declarativos que representan las proposiciones atómicas se corresponden con la realidad, decimos que son verdaderos; mientras que cuando este no es el caso decimos que son falsos.

Si combinamos los enunciados declarativos p y q con un conectivo lógico como $\wedge$, entonces el valor de verdad de $p \wedge q$ está determinado por tres aspectos: el valor de verdad de p, el valor de verdad de q y el significado de $\wedge$. El significado de la conjunción captura la observación de que $p \wedge q$ es verdadera si y sólo si (ssi) p y q son ambas verdaderas; de otra forma $p \wedge q$ es falsa. De forma que, desde la perspectiva de $\wedge$ todo lo que debemos saber es si p y q son verdaderos. No es necesario saber que es lo que dicen p y q acerca del mundo. Esto también es el caso para el resto de los operadores lógicos y es la razón por la cual podemos computar el valor de verdad de una expresión con solo saber el valor de verdad de las proposiciones atómicas que ocurren en ella. Formalicemos estos conceptos:

Definición 4.2 (Valores de verdad)**.** *El conjunto de valores de verdad contiene dos elementos T y F donde T representa verdadero, y F falso.*

Definición 4.3 (Modelo)**.** *Un modelo o valuación de una fórmula ϕ es una asignación de valores de verdad a las proposiciones atómicas que ocurren en ϕ.*

Ejemplo 4.4. *La función que asigna $T \mapsto q$ y $F \mapsto p$ es un modelo de la fórmula $p \vee \neg q$.*

Podemos pensar que el significado de la conjunción es una función de dos argumentos. Cada argumento es un valor de verdad y el resultado de la función es también un valor de verdad. Podemos especificar esto en una tabla, llamada **tabla de verdad** de la conjunción (Ver cuadro 4.1). *Tabla de verdad*

La tabla de verdad de la disyunción se muestra en el cuadro 4.2. La de la implicación se muestra en el cuadro 4.3.

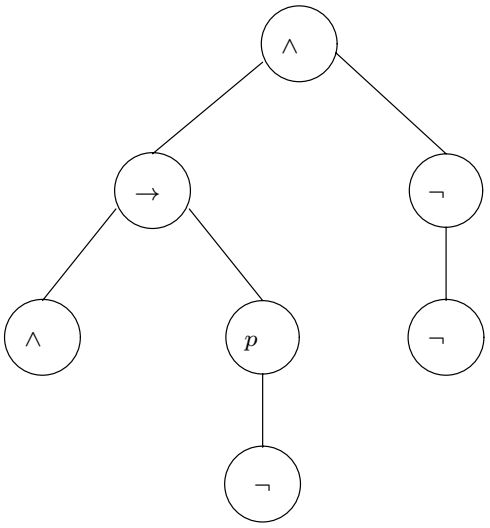

Figura 4.3: El árbol sintáctico de una expresión que no es una fórmula bien formada.

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

Cuadro 4.1: La tabla de verdad de la conjunción ($\wedge$).

Observen que, como cada argumento puede tener dos valores de verdad, el número de combinaciones posibles es 2^n donde n es el número de argumentos del operador. Por ejemplo, $\phi \wedge \psi$ tiene $2^2 = 4$ casos a considerar, los que se listan en la tabla de verdad del cuadro 4.1. Pero $\phi \wedge \psi \wedge \chi$ tendría $2^3 = 8$ casos a considerar.

La semántica de la implicación (Cuadro 4.3) es poco intuitiva. Piensen en ella como una relación que preserva la verdad. Es evidente que $T \rightarrow F$ no preserva la verdad puesto que inferimos algo que es falso a partir de algo que es verdadero. Por ello, la entrada correspondiente en la tabla de verdad de la implicación da como salida falso. También es evidente que $T \rightarrow T$ preserva la verdad; pero los casos donde el primer argumento tiene valor de verdad F también lo hacen, porque no tienen verdad a preservar dado que el supuesto de la implicación es falso.

Si la semántica de la implicación sigue siendo incomoda, consideren que, al menos en cuanto a los valores de verdad se refiere, la expresión $\phi \rightarrow \psi$ es una abreviatura de $\neg\phi \vee \psi$. Las tablas de verdad pueden usarse para probar que cuando la primer expresión mapea a verdadero, también lo hace la segunda. Esto quiere decir que ambas expresiones son **semánticamente equivalentes**. Aunque claro, las reglas de la deducción natural tratan a ambas fórmulas de manera muy diferente, dado que su sintaxis es bien diferente.

Equivalencia semántica

La tabla de verdad de la negación se muestra en el cuadro 4.4. Observen que en este caso tenemos $2^1 = 2$ casos que considerar. También pueden definirse tablas de verdad para la contradicción y su negación: $\bot \mapsto F$ y $\top \mapsto T$.

Ejemplo 4.5. *¿Cual es el valor de verdad de la expresión $\neg p \wedge q \rightarrow p \wedge (q \vee \neg r)$, si el modelo es $F \mapsto q$, $T \mapsto p$ y $T \mapsto r$? Una de las ventajas de nuestra semántica es que es **composicional**. Si sabemos los valores de verdad de $\neg p \wedge q$ y $p \wedge (q \vee \neg r)$, entonces podemos usar la tabla de verdad de la implicación para saber el valor de verdad de toda la expresión. De tal forma que podemos recorrer el árbol sintáctico de la expresión en forma ascendente para computar su valor de verdad. Primero, sabemos el valor de las hojas dado*

Semántica composicional

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

Cuadro 4.2: La tabla de verdad de la disyunción ($\vee$).

ϕ	ψ	$\phi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

Cuadro 4.3: La tabla de verdad de la implicación ($\rightarrow$).

el modelo inicial. Puesto que el valor de verdad de p es verdadero, el valor de $\neg p$ es falso, q es falsa y la conjunción de falso y falso da falso. De forma que el lado izquierdo de la conjunción evalúa a falso. Para el lado derecho tenemos que el modelo inicial asigna a r el valor de verdad verdadero, de forma que $\neg r$ es falso. La disyunción de q que tiene un valor falso, con falso, da falso. Y la conjunción de p, cuyo valor es verdadero con falso, da falso. Finalmente, la implicación de dos valores de verdad falsos, da verdadero. La figura 4.4 ilustra la propagación de valores de verdad por este árbol sintáctico, como se ha explicado en el ejemplo.

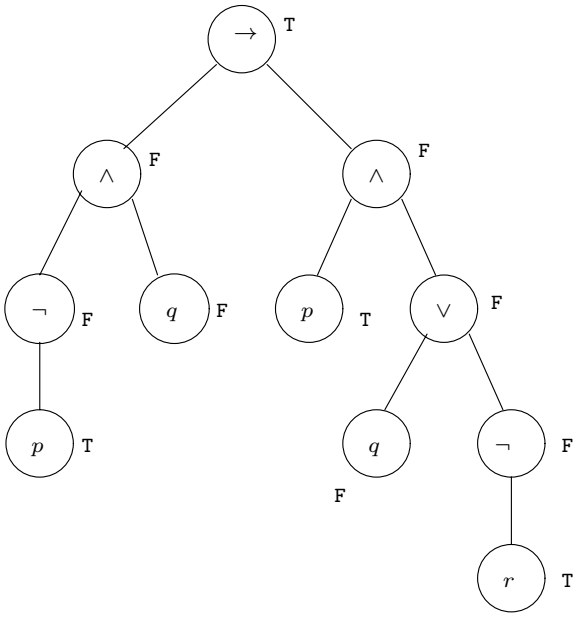

Figura 4.4: La evaluación de una fórmula lógica a partir de su árbol sintáctico, dado un modelo inicial.

Ejemplo 4.6. *Construya la tabla de verdad de la expresión* $(p \rightarrow \neg q) \rightarrow (q \vee \neg p)$*:*

ϕ	$\neg\phi$
T	F
F	T

Cuadro 4.4: La tabla de verdad de la negación ($\neg$).

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$(p \to \neg q) \to (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

4.2 INDUCCIÓN MATEMÁTICA

A la edad de ocho años, Gauss se aburría en clase y no prestaba la atención debida a su profesor. Como es de imaginarse, al pequeño se le impuso un castigo que consistía en sumar los números del uno al cien. Un burdo rumor dice que Gauss [3] respondió 5050 a los pocos segundos de haber sido castigado, para sorpresa de su profesor ¿Cómo logró Gauss librarse tan fácilmente del castigo? Quizá sabía que:

$$1+2+3+4+\cdots+n = \frac{n(n+1)}{2} \tag{4.2}$$

de forma que:

$$\begin{aligned} 1+2+3+4+\cdots+100 &= \frac{100(101)}{2} \\ &= 5050 \end{aligned}$$

Inducción matemática

La **inducción matemática** nos permite probar que la ecuación 4.2 se sostiene para valores arbitrarios de n. De manera más general, podemos decir que nos permite probar que **todo** número natural satisface cierta propiedad. Supongamos que tenemos una propiedad M que creemos es verdadera para todo número natural. Escribimos $M(5)$ para denotar que la propiedad es verdadera para 5. Supongamos ahora que sabemos lo siguiente de M:

1. **Caso base:** El número natural 1 tiene la propiedad M, es decir, tenemos una prueba de que $M(1)$.
2. **Paso inductivo:** Si n es un número natural que asumimos tiene la propiedad $M(n)$, entonces podemos probar que $n+1$ tiene la propiedad $M(n+1)$; es decir, tenemos una prueba de que $M(n) \to M(n+1)$.

Definición 4.4. *El principio de inducción matemática dice que, basados en estas dos piezas de información, todo número natural n tiene la propiedad $M(n)$. Al hecho de suponer $M(n)$ en el paso inductivo, se le conoce como* ***hipótesis de la inducción****.*

¿Porqué tiene sentido este principio? Tomemos cualquier número natural k. Si k es igual a 1, entonces usando el caso base podemos probar que k tienen la propiedad $M(1)$. En cualquier otro caso, podemos usar el paso inductivo aplicado a $n = 1$ para inferir que $2 = 1+1$ tiene la propiedad $M(2)$. Podemos hacer esto usando la eliminación de la implicación ($\to e$) puesto que sabemos que 1 tiene la propiedad en cuestión. Igualmente podemos probar $M(3)$ y así hasta llegar a k. Regresemos al ejemplo de Gauss.

3 Lo que no es un rumor, es su precocidad: terminó sus *Disquisitiones Arithmeticae* a los 21 años.

Teorema 4.1. *La suma de* $1+2+3+4+\cdots+n$ *es igual a* $n(n+1)/2$ *para todo número natural* n*.*

La prueba es como sigue: Denotamos por LIE_n el lado izquierdo de la igualdad, $1+2+3+4+\cdots+n$; y por LDE_n el lado derecho de la igualdad $n(n+1)/2$ para todo $n \geq 1$.

- **Caso base:** Si $n = 1$ entonces $LIE_1 = 1$ (solo hay un sumando), que es igual a $LDE_1 = 1(1+1)/2 = 1$.
- **Paso inductivo:** Asumamos que $LIE_n = LDE_n$. Recuerden que este supuesto es llamado hipótesis inductiva; es la guía de nuestro argumento. Necesitamos probar que $LIE_{n+1} = LDE_{n+1}$, esto es, que $1+2+3+4+\cdots+(n+1)$ es igual que $(n+1)(((n+1)+1)/2)$. La clave está en observar que la suma de $1+2+3+4+\cdots+(n+1)$ no es otra cosa que la suma de dos sumandos $1+2+3+4+\cdots+n$ y $(n+1)$; y que el primer sumando es nuestra hipótesis inductiva, de forma que:

$$\begin{aligned}
LIE_{n+1} &= 1+2+3+4+\cdots+n+(n+1) \\
&= LIE_n + (n+1) \quad \text{reagrupando la suma} \\
&= LDE_n + (n+1) \quad \text{por la hipótesis inductiva} \\
&= \frac{n(n+1)}{2} + (n+1) \\
&= \frac{n(n+1)}{2} + \frac{2(n+1)}{2} \\
&= \frac{(n+2)(n+1)}{2} \\
&= \frac{(n+1)(n+2)}{2} \\
&= \frac{(n+1)((n+1)+1)}{2} \\
&= LDE_{n+1}
\end{aligned}$$

De forma que, como hemos probado el caso base y el paso inductivo, podemos inferir matemáticamente que todo número natural n satisface el teorema anterior. □

Existe una variante de inducción matemática en la que la hipótesis inductiva para probar $M(n+1)$ no es solo $M(n)$, sino la conjunción $M(1) \wedge M(2) \wedge \ldots \wedge M(n)$. En esta variante, llamada **curso de valores**, no es necesario tener un caso base explícito, todo puede hacerse en el paso inductivo. *Curso de valores*

¿Cómo puede funcionar la inducción sin un caso base? La respuesta es que el caso base se incluye implicitamente en el paso inductivo. Consideren el caso de $n = 3$: el paso inductivo es $M(1) \wedge M(2) \wedge M(3) \rightarrow M(4)$. Ahora consideren el caso $n = 1$ entonces el paso inductivo es $M(1) \rightarrow M(2)$ ¿Qué sucede cuando el caso es $n = 0$? En ese caso no hay fórmulas a la izquierda de la implicación, por lo que se debe probar $M(1)$ a partir de nada. Veamos algunos ejemplos.

En las Ciencias de la Computación solemos trabajar con estructuras finitas de algún tipo: estructuras de datos, programas, archivos, etc. En muchas ocasiones debemos probar que toda ocurrencia de una estructura de datos tiene cierta propiedad. Por ejemplo, la definición 4.1 de fórmula bien formada tiene la propiedad de que el número de paréntesis que abren es igual al de los paréntesis que cierran. Podemos usar la inducción matemática en el dominio de los números naturales para probar esto. Para tener éxito, debemos ligar de alguna forma las fórmulas bien formadas con los números naturales.

Definición 4.5. *Dada una fórmula bien formada* ϕ*, definimos su* ***altura*** *como 1 más la longitud de la rama más larga del árbol.*

Por ejemplo, la rama más larga del árbol que se muestra en la figura 4.5 es de longitud 4 por lo que su altura es 5. Observen que la altura de un átomo proposicional es $1 + 0 = 1$. Puesto que toda fórmula bien formada tiene una altura finita, podemos demostrar enunciados sobre las fórmulas bien formadas haciendo inducción matemática sobre su altura. Este truco suele conocerse como **inducción estructural**, una importante técnica de razonamiento de Ciencias de la Computación. Usando la noción de altura de un árbol sintáctico nos damos cuenta de que la inducción estructural es solo un caso especial de la inducción por cursos-de-valores.

Inducción estructural

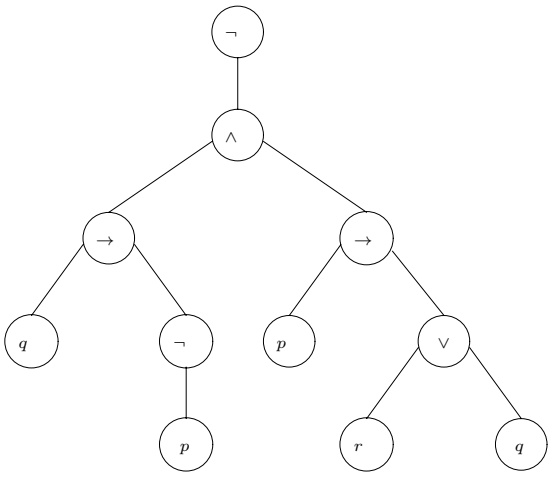

Figura 4.5: Un árbol sintáctico de altura 5.

Teorema 4.2. *Para toda fórmula bien formada de la lógica proposicional, el número de paréntesis que abre es igual al número de paréntesis que cierran.*

Prueba: Procederemos por inducción por curso de valores sobre la altura del árbol sintáctico de las fórmulas bien formadas ϕ . Denotemos por $M(n)$ que "todas las fórmulas de altura n tienen el mismo número de paréntesis que abren y cierran." Asumimos $M(k)$ para cada $k < n$ y tratamos de probar $M(n)$. Tomemos una fórmula ϕ de altura n.

- **Caso base:** Cuando $n = 1$, ϕ es un átomo proposicional por lo que no hay paréntesis en la expresión y $M(1)$ se satisface: $0 = 0$.
- **Paso inductivo por curso-de-valores:** Cuando $n > 1$ la raíz del árbol sintáctico de ϕ debe ser $\neg$, $\rightarrow$, $\vee$ o $\wedge$. Supongamos que es $\rightarrow$ (los otros tres casos se argumentan de manera similar). Entonces ϕ es igual a $(\phi_1 \rightarrow \phi_2)$ para las fórmulas bien formadas ϕ_1 y ϕ_2. Es claro que la altura de los árboles sintácticos de estas dos expresiones es menor que n. Usando la hipótesis de inducción concluímos que ϕ_1 tiene el mismo número de paréntesis que abren y cierran; lo mismo que ϕ_2. Pero en $(\phi_1 \rightarrow \phi_2)$ agregamos un paréntesis que abre y uno que cierra por lo que el número de ocurrencias de los paréntesis en ϕ es el mismo. □

4.3 ROBUSTEZ DE LA LÓGICA PROPOSICIONAL

Las reglas de la deducción natural permiten desarrollar rigurosos hilos de argumentación a través de los cuales concluímos que ψ asumiendo otras proposiciones como $\phi_1, \phi_2, \ldots, \phi_n$. En ese caso decimos que la inferencia $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es válida ¿Tenemos evidencia de que las reglas son todas **correctas** en el sentido de que preserven la verdad computada con nuestras semántica basada en tablas de verdad?

Robustez

Dada una prueba de $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ ¿Es concebible que exista un modelo donde ψ es falso aunque las proposiciones $\phi_1, \phi_2, \ldots, \phi_n$ sean verdaderas? Afortunadamente este no es el caso y en esta sección lo demostraremos formalmente. Supongamos que tenemos una prueba en nuestro cálculo de deducción natural que establece que la inferencia $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es válida. Necesitamos probar que para todo modelo donde las proposiciones $\phi_1, \phi_2, \ldots, \phi_n$ son verdaderas, ψ también lo es.

Definición 4.6. *Si, para todos los modelos donde $\phi_1, \phi_2, \ldots, \phi_n$ son verdaderas, ψ también lo es, entonces decimos que:*

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

*El símbolo $\models$ se llama relación de **consecuencia lógica** o vinculación lógica (*logical entailment*).*

Veamos algunos ejemplos de esta noción:

1. ¿Es el caso que $p \wedge q \models p$? Para responder debemos inspeccionar todas las asignaciones de verdad para p y q. Cuando la asignación de valores compute T para $p \wedge q$ debemos asegurarnos de que ese también es el caso para p. Pero $p \wedge q$ solo computa T cuando p y q son verdaderas, por lo que p es consecuencia lógica de $p \wedge q$.

2. ¿Qué hay acerca de $p \vee q \models p$? Hay tres asignaciones de verdad donde $p \vee q$ es T, de forma que p debería ser T en todas ellas. Sin embargo, si asignamos T a q y F a p la disyunción computa T pero p es falsa. De forma que la relación $p \vee q \models p$ no se mantiene.

3. ¿Qué sucede si modificamos la inferencia anterior para que sea $p \vee q \models p$. Observe que esto obliga a focalizar en evaluaciones donde q es falsa, lo cual obliga a que p sea verdadera si queremos que la disyunción lo sea. Por tanto, es el caso que $p \vee q \models p$

4. Observen que $p \models q \vee \neg q$ se da, aún cuando no existen ocurrencias de las proposiciones atómicas del antecedente en el consecuente.

De la discusión anterior podemos adivinar que el argumento de la robustez consiste en probar que si $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$, entonces se da que $\phi_1, \phi_2, \ldots \phi_n \models \psi$.

Teorema 4.3 (Robustez)**.** *Sean ϕ_1, ϕ_2, ϕ_n y ψ fórmulas lógicas proposicionales. Si la inferencia $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$ es valida, entonces es el caso que $\phi_1, \phi_2, \ldots \phi_n \models \psi$.*

Prueba: Puesto que $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$ es válida, conocemos una prueba de ψ a partir de las premisas $\phi_1, \phi_2, \ldots \phi_n$. Ahora aplicamos un truco estructural, razonaremos por **inducción matemática sobre la longitud de esta prueba**. La longitud de una prueba es el número de líneas en ella. De forma que intentamos mostrar $M(k)$: para toda consecuencia $\phi_1, \phi_2, \ldots \phi_n \vdash \psi \quad (n \geq 0)$ que tiene una prueba de longitud k, es el caso que $\phi_1, \phi_2, \ldots \phi_n \models \psi$ se da. Veamos:

La consecuencia $p \wedge q \rightarrow r \vdash p \rightarrow (q \rightarrow r)$ tiene una prueba:

1.	$p \wedge q \rightarrow r$	*premisa*
2.	p	*supuesto*
3.	q	*supuesto*
4.	$p \wedge q$	$(\wedge i)$ 2, 3
5.	r	$(\rightarrow e)$ 1, 4
6.	$q \rightarrow r$	$(\rightarrow i)$ $3-5$
7.	$p \rightarrow (q \rightarrow r)$	$(\rightarrow i)$ $2-6$

Si eliminamos las dos últimas líneas ya no tenemos una prueba, pues la caja más externa quedaría sin cerrar. Sin embargo, podemos obtener una prueba removiendo la última línea y escribiendo el supuesto de la caja más externa como una premisa:

1.	$p \wedge q \to r$	*premisa*
2.	p	*premisa*
3.	q	*premisa*
4.	$p \wedge q$	$(\wedge i)$ 2, 3
5.	r	$(\to e)$ 1, 4
6.	$q \to r$	$(\to i)$ $3-5$

Esto es una prueba de la inferencia $p \wedge q \to r, p \vdash q \to r$. La hipótesis inductiva garantiza entonces que $p \wedge q \to r, p \models q \to r$. Entonces podemos razonar que también es el caso que $p \wedge q \to r \models p \to (q \to r)$ ¿Porqué?

Procedamos con la prueba por inducción. Asumamos que $M(k')$ para cada $k' < k$ y tratemos de probar que $M(k)$.

- **Caso base:** una prueba de longitud 1. Si $k = 1$ entonces la prueba debe ser de la forma:

  1. ϕ premisa

  puesto que todas las demas reglas involucran más de una linea. Este es el caso cuando $n = 1$ y ϕ_1 y ψ son iguales a ϕ, es decir, la inferencia es $\phi \vdash \phi$. Evidentemente si ϕ evalua a T, es el caso que $\phi \models \phi$.

- **Paso inductivo por curso de valores:** Asumamos que la prueba de la consecuencia $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ tiene una longitud k y que el enunciado que queremos probar es verdadero para todo número menor que k. Nuestra prueba tiene la siguiente estructura:

  1. ϕ_1 premisa
  2. ϕ_2 premisa
  ⋮
  n. ϕ_n premisa
  ⋮
  k. ψ justificación

  Hay dos cosas que no sabemos en este punto. Primero, ¿Qué está pasando en los puntos suspensivos? Segundo, ¿Cual fue la última regla aplicada? Es decir, ¿Cual fue la justificación de la última línea? El primer punto no es de nuestro interés, es aquí que la inducción matemática muestra su poder. El segundo punto es donde todo el trabajo de esta prueba descansa. En general, no hay una forma simple de saber que regla fue aplicada a lo último, por lo que necesitamos considerar todos los casos posibles:

  1. Supongamos que la última regla es $(\wedge i)$, entonces sabemos que ψ tiene la forma $\psi_1 \wedge \psi_2$ y la justificación de la línea k hace referencia a dos líneas precedentes que tienen a ψ_1 y ψ_2 respectivamente, como conclusiones. Supongamos que esas líneas son k_1 y k_2. Dado que k_1 y k_2 son menores que

k observamos que existen pruebas de las inferencias $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1$ y $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_2$ con una longitud menor que k. Usando la hipótesis inductiva concluimos que $\phi_1, \phi_2, \ldots, \phi_n \models \psi_1$ y $\phi_1, \phi_2, \ldots, \phi_n \models \psi_2$. Estas dos relaciones implican que $\phi_1, \phi_2, \ldots, \phi_n \models \psi_1 \wedge \psi_2$.

2. Supongamos que la última regla para demostrar ψ es $(\vee e)$, entonces debimos probar, asumiendo o adoptando la premisa de que alguna fórmula $\eta_1 \vee \eta_2$ en alguna línea k' dond $k' < k$ que es referenciada por $(\vee e)$ en la línea k. Por lo tanto, tendremos una prueba más corta de la consecuencia $\phi_1, \phi_2, \ldots \phi_n \vdash \eta_1 \vee \eta_2$, obtenida al convertir los supuestos de las cajas que se abren en la linea k' en premisas. De forma similar obtenemos pruebas de las consecuencias $\phi_1, \phi_2, \ldots \phi_n, \eta_1 \vdash \psi$ y $\phi_1, \phi_2, \ldots \phi_n, \eta_2 \vdash \psi$. Por la hipótesis inductiva concluímos que $\phi_1, \phi_2, \ldots \phi_n \models \eta_1 \vee \eta_2$, $\phi_1, \phi_2, \ldots \phi_n, \eta_1 \models \psi$ y $\phi_1, \phi_2, \ldots \phi_n, \eta_2 \models \psi$. En su conjunto, estas tres relaciones hacen que $\phi_1, \phi_2, \ldots \phi_n \models \psi$.
3. La argumentación continua probando que todas las reglas de prueba se comportan semánticamente en la misma forma que las tablas de verdad correspondiente. □

No existencia de demostración

La robustez de la lógica proposicional es útil para garantizar la **no existencia** de una prueba para una consecuencia dada. Digamos que queremos probar que $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es válida, pero no lo conseguimos ¿Cómo podemos estar seguros de que no hay una prueba para ese caso? Basta con encontrar un modelo en donde ϕ_i evalua a T mientras que ψ evalua a F. Entonces por la definición de consecuencia lógica, no es el caso que $\phi_1, \phi_2, \ldots \phi_n \models \psi$ y, usando la robustez, esto significa que la consecuencia $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ no es válida; y por tanto, no hay una prueba para ella.

4.4 COMPLETITUD DE LA LÓGICA PROPOSICIONAL

Completitud

En esta sección probaremos que las reglas de la deducción natural de la lógica proposicional son **completas**. Cuando es el caso que $\phi_1, \phi_2, \ldots, \phi_n \models \psi$, entonces existe una prueba de deducción natural para la consecuencia $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$. Combinando esto con el resultado anterior obtenemos que $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es válida, si y sólo si $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

Búsqueda de prueba

Esto nos da cierta libertad con respecto al método que preferiremos usar. El primer método involucra una **búsqueda de prueba**, que es la base del paradigma de la **programación lógica**. El segundo método nos obliga a computar una tabla de verdad que es exponencial en el número de proposiciones atómicas involucradas en una fórmula. Ambos métodos son intratables en lo general, pero ciertas fórmulas particulares responden diferente a ellos.

El argumento que construiremos en esta sección se da en tres pasos:

1. Mostraremos que $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi)))$ es el caso.
2. Mostraremos que $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi)))$ es válida.
3. Finalmente, mostraremos que $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es válida.

El primer paso y el tercero son bastante fáciles, todo el trabajo real se concentra en el segundo paso.

4.4.1 Paso 1

Definición 4.7 (Tautología)**.** *Una fórmula de la lógica proposicional ϕ es llamada una tautología si y sólo si evalua T bajo cualquier modelo. Es decir, si $\models \phi$.*

Supongamos que $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ es el caso. Verifiquemos que $\phi_1 \rightarrow (\phi_2 \rightarrow (\ldots(\phi_n \rightarrow \psi)))$ es una tautología. Como la fórmula en cuestión es una implicación, solo puede evaluar F si y sólo si todas las ϕ_i evaluan a T y ψ evalua a F. Pero esto contradice el hecho de que $\phi_1, \phi_2, \ldots, \phi_n \models \psi$; por lo tanto $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots(\phi_n \rightarrow \psi)))$.

4.4.2 Paso 2

Definición 4.8. *Si $\models \eta$, entonces $\vdash \eta$ es válida. En otras palabras, si η es una tautología, entonces η es un teorema.*

Asumamos que $\models \eta$. Dado que η contiene n distintos átomos proposicionales $p_1, p_2, \ldots, p_n$ sabemos que η es T para todas las 2^n lineas de su tabla de verdad. Cada línea lista un modelo de η ¿Cómo podemos usar esta información para construir una prueba de η? En algunos casos esto puede hacerse facilmente, observando cuidadosamentela estructura de η, pero aquí deberemos contender con una forma **uniforme** de construir tal prueba. La clave está en codificar cada linea de la tabla de verdad de η como una consecuencia. Entonces construiremos pruebas para las 2^n inferencias y las ensamblaremos en la prueba de η.

Proposición 4.1. *Sea ϕ una fórmula tal que $p_1, p_2, \ldots, p_n$ son sus únicos átomos proposicionales. Sea l cualquier número de línea en la tabla de verdad de ϕ. Para todo $1 \leq i \leq n$, $\hat{p}_i$ es p_i si la entrada de la línea l de p_i es T; en cualquier otro caso $\hat{p}_i$ es $\neg p_i$. Entonces tenemos:*

1. *$\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi$ es demostrable si la entrada para ϕ en la línea l es verdadera.*
2. *$\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\phi$ es demostrable si la entrada para ϕ en la línea l es falsa.*

Prueba: Esta prueba se lleva a cabo por inducción estructural sobre la fórmula ϕ, que es una inducción matemática sobre la altura del árbol sintáctico de ϕ.

- Si ϕ es un átomo proposicional p, debemos mostrar que $p \vdash p$ y que $\neg p \vdash \neg p$. Se puede hacer directamente la prueba.
- Si ϕ es de la forma $\neg\phi_1$, nuevamente tenemos dos casos a considerar. Primero, asumamos que ϕ es verdadera. En ese caso ϕ_1 es falsa. Observen que ϕ_1 tiene las mismas proposiciones atómicas que ϕ. Debemos usar la hipótesis de inducción sobre ϕ_1 para concluir que $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\phi_1$, pero $\neg\phi_1$ es ϕ; Segundo si ϕ es falsa, entonces ϕ_1 es verdadera y tenemos que, por inducción $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1$. Usando $(\neg\neg i)$ podemos extener esta prueba en $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\neg\phi_1$. Pero $\neg\neg\phi_1$ es $\neg\phi$.

El resto de los casos trabajan con dos sub-fórmulas: ϕ es igual a $\phi_1 \circ \phi_2$, donde $\circ$ es $\rightarrow$, $\wedge$ ó $\vee$. En todos estos casos, sea $q_1, q_2, \ldots, q_l$ los átomos proposicionales en ϕ_1 y $r_1, r_2, \ldots, r_k$ los átomos proposicionales en ϕ_2. Entonces la siguiente igualdad se satisface $\{q_1, q_2, \ldots, q_l\} \cup \{r_1, r_2, \ldots, r_k\} = \{p_1, \ldots, p_n\}$. De forma que siempre que $\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_l \vdash \phi_1$ y $\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_k \vdash \phi_2$ son válidas, también lo es $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1 \wedge \phi_2$ usando la regla de $(\wedge i)$.

- Sea ϕ de la forma $\phi_1 \rightarrow \phi_2$. Si ϕ es falsa, entonces sabemos que ϕ_1 también lo es, mientras que ϕ_2 es verdadera. Usando nuestra hipótesis inductiva tenemos que $\hat{q}_1, \ldots, \hat{q}_l \vdash \phi_1$ y que $\hat{r}_1, \ldots, \hat{r}_k \vdash \neg\phi_2$. De forma que $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1 \wedge \neg\phi_2$. El resto consiste entonces en probar que la inferencia $\phi_1 \wedge \neg\phi_2 \vdash \neg(\phi_1 \rightarrow \phi_2)$ que queda como ejercicio sugerido. Si ϕ es verdadero entonces hay tres casos a considerar. Primero, si ϕ_1 y ϕ_2 son falsos. Por hipótesis inductiva tenemos que: $\hat{q}_1, \ldots, \hat{q}_l \vdash \neg\phi_1$ y $\hat{r}_1, \ldots, \hat{r}_k \vdash \neg\phi_2$. De forma que $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\phi_1 \wedge \neg\phi_2$. Solo queda probar que la inferencia $\neg\phi_1 \wedge \neg\phi_2 \vdash \phi_1 \rightarrow \phi_2$ es válida. Segundo, si ϕ_1 es falso y ϕ_2 es verdadero usamos la hipótesis inductiva para llegar

a $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\phi_1 \wedge \phi_2$ y tendríamos que probar que $\neg\phi_1 \wedge \phi 2 \vdash \phi_1 \rightarrow \phi_2$ es una inferencia válida. Tercero, si ϕ_1 y ϕ_2 son verdaderos, llegamos a $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1 \wedge \phi_2$ y solo resta probar que $\phi_1 \wedge \phi 2 \vdash \phi_1 \rightarrow \phi_2$.

Veamos un ejemplo de cómo funcionan estas pruebas basadas en la tabla de verdad de una consecuencia. Si queremos probar de esta manera que la fórmula bien formada $p \wedge q \rightarrow p$ es una tautología, tendremos que considerar que las cuatro líneas (2^2) de su tabla de verdad deberían ser verdaderas. Por ello tendríamos cuatro pruebas del tipo $\hat{p_1}, \hat{p}_2 \vdash \eta$:

$$
\begin{aligned}
p, q &\vdash p \wedge q \rightarrow p \\
\neg p, q &\vdash p \wedge q \rightarrow p \\
p, \neg q &\vdash p \wedge q \rightarrow p \\
\neg p, \neg q &\vdash p \wedge q \rightarrow p
\end{aligned}
$$

Puesto que queremos probar que esta proposición es una tautología, debemos encargarnos de que la demostración de que es un teorema no tenga premisas. Para contender con el lado izquierdo de las cuatro consecuencias, recurriremos al *LEM*. La prueba es como sigue:

1.	$p \vee \neg p$	*LEM*
2.	p	*supuesto*
3.	$q \vee \neg q$	*LEM*
4.	q	*supuesto*
5.	$\vdots$	$\vdots$
6.	$p \wedge q \rightarrow p$	
7.	$\neg q$	*supuesto*
8.	$\vdots$	$\vdots$
9.	$p \wedge q \rightarrow p$	
10.	$p \wedge q \rightarrow p$	$(\vee e) 3, 4-6, 7-9$
11.	$\neg p$	*supuesto*
12.	$q \vee \neg q$	*LEM*
13.	q	*supuesto*
14.	$\vdots$	$\vdots$
15.	$p \wedge q \rightarrow p$	
16.	$\neg q$	*supuesto*
17.	$\vdots$	$\vdots$
18.	$p \wedge q \rightarrow p$	
19.	$p \wedge q \rightarrow p$	$(\vee e) 12, 13-15, 16-18$
20.	$p \wedge q \rightarrow p$	$(\vee e) 1, 2-10, 11-19$

4.4.3 Paso 3

Finalmente, necesitamos encontrar una prueba de que $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ es una inferencia válida. Tomamos la prueba de que $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots(\phi_n \rightarrow \psi)))$ obtenida en el paso 2 y aumentamos la prueba introduciendo $\phi_1, \ldots, \phi_n$ como premisas. Entonces aplicamos $(\rightarrow e)$ n veces sobre cada una de las premisas y llegaremos a la conclusión que ψ lo cual nos da la prueba buscada.

Corolario 4.1 (Robustez y Completitud)**.** *Sean $\phi_1, \ldots, \phi_n$ y ψ fórmulas de la lógica proposicional. Entonces $\phi_1, \ldots \phi_n \models \psi$ si y sólo si $\phi_1, \ldots \phi_n \vdash \psi$ es válida.*

4.5 FORMAS NORMAL CONJUNTIVA

En la sección anterior probamos que nuestro sistema de prueba para la lógica proposicional es robusto y completo para la semántica basada en tablas de verdad. Esto quiere decir que todo lo que probemos es un hecho verdadero; y que, toda consecuencia válida tiene una prueba en el sistema de deducción natural. Esta conexión nos permitirá movernos libremente entre el nivel de prueba ($\vdash$) y el de consecuencia lógica ($\models$). En lo qu sigue exploraremos formas alternativas de probar que $\phi_1, \phi_2, \phi_n, \ldots, \models \psi$ basados en la transformación sintáctica de estas fbfs en equivalentes sintácticos que puedan resolverse algoritmicamente. Esto requiere definir con precisión el concepto de **equivalencia**.

Equivalencia semántica

Definición 4.9 (Equivalencia semántica)**.** *Sean ϕ y ψ fbfs de la lógica proposicional. $\phi \equiv psi$, ϕ es semátincamente equivalente a ψ ssi $\phi \models \psi$ y $\psi \models \phi$.*

Decimos que ϕ es válida, ssi $\models \phi$. También pudimos definir $\phi \equiv \psi$ para denotar que $\models (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$. Dadas la robustez y la completitud, la equivalencia semántica y de prueba son idénticas.

Ejemplo 4.7. *Las siguientes equivalencias semáticas son válidas:*

- $p \rightarrow q \equiv \neg q \rightarrow \neg p$
- $p \rightarrow q \equiv \neg p \vee q$
- $p \wedge q \rightarrow p \equiv r \vee \neg r$
- $p \wedge q \rightarrow r \equiv p \rightarrow (q \rightarrow r)$

En lo que sigue, necesitaremos una prueba de que todo procedmiento de decisión para tautologías, lo es también para consecuencias válidas:

Lema 4.1. *Dadas la fórmulas $\phi_1, \phi_2, \ldots, \phi_n$ y ψ de la lógica proposicional, $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ ssi $\models \phi_1 \rightarrow (\phi_2 \rightarrow \ldots \rightarrow (\phi_n \rightarrow \psi))$.*

Su prueba es como sigue: Suponemos que $\models \phi_1 \rightarrow (\phi_2 \rightarrow \ldots \rightarrow (\phi_n \rightarrow \psi))$ es el caso. Si las ϕ_i son verdaderas bajo una valuación, también lo debe ser ψ. A la inversa, es el paso 1 de la prueba de completitud. □

Definición 4.10 (Forma Normal Conjuntiva)**.** *Una literal L es un átomo p o su negación (4.3). Una fórmula C está en **Formal Normal Conjuntiva** (CNF) si es una conjunción de cláusulas (4-5), donde cada cláusula es una disyunción de literales (4.4).*

Forma Normal Conjuntiva

$$L ::= p \mid \neg p \tag{4.3}$$
$$D ::= L \mid L \vee D \tag{4.4}$$
$$C ::= D \mid D \wedge C \tag{4.5}$$

Ejemplo 4.8. *Las siguientes fbfs están en forma nornal conjuntiva:*

- $(\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q$
- $(p \vee r) \wedge (\neg p \vee r) \wedge (p \vee \neg r)$

La **relevancia** de la CNF es que permite verificar la validez de una fbf fácilmente, proceso que de otra forma es exponencial en el número de átomos de la fbf a verificar. *Relevancia*

Lema 4.2. *Una disyunción de literales $L_1 \vee ... \vee L_m$ es válida ssi existe un $L_i = \neg L_j$ para $1 \leq i, j \leq m$.*

Definición 4.11 (Satisfacción)**.** *Sea ϕ una fbf proposicional, ϕ es **satisfacible** ssi $\neg\phi$ no es válida.* *Satisfacción*

La prueba es como sigue: Asumimos que ϕ es satisfacible. Por definición, existe una valuación donde ϕ es verdadera, pero eso implica que $\neg\phi$ sería falsa en la misma valuación, por lo que $\neg\phi$ no pude ser válida; Asumimos que $\neg\phi$ no es valida, debe haber una valuación donde $\neg\phi$ sea falsa, en cuyo caso ϕ es verdadera y por tanto satisfacible. □

Este resultado es muy intresante, pues implica que solo necesitamos un procedimiento de decisión para ambos conceptos.

4.5.1 Conversión de fbf a CNF

La ganancia en la eficiencia del procedimiento de prueba, se paga con el precio de convertir una fbf proposicional a su equivalente en CNF. Comenzaremos por un algoritmo **determinista** que siempre computa la misma CNF para una fbf ϕ de entrada. Sus características deseables incluyen:

1. CNF termina para toda fbfb de la lógica proposicional.
2. Para cada fbf de la lógica proposicional CNF computa una fbf equivalente.
3. La fbf resultante esta en forma normal conjuntiva.

La **estrategia** a seguir conciste en proceder por inducción estructural sobre la fbf ϕ. La inducción estructural garantiza las características deseables del algoritmo. *Estrategia*

Ejemplo 4.9. *Si ϕ es de la forma $\phi_1 \wedge \phi_2$ computar la CNF η_1 para ϕ_i, $i = 1, 2$; de forma que $\eta_1 \wedge \eta_2$ es la CNF equivalente a ϕ.*

De forma que la CNF se resuelve por casos:

1. Si ϕ es una literal, por definición está en CNF y la salida es ϕ.
2. Si ϕ es de la forma $\phi_1 \wedge \phi_2$ se llama a CNF recursivamente sobre cada ϕ_i para obtener η_1 y η_2. La CNF es $\eta_1 \wedge \eta_2$.
3. Si ϕ tiene la forma $\phi_1 \vee \phi_2$ se llama a CNF sobre cada ϕ_i, pero no regresamos $\eta_1 \vee \eta_2$ puesto que esta solo es una forma normal si η_i son literales.
4. Se debe distribuir la disyunción sobre la conjunción, garantizando que la disyunciones generadas sean de literales. Una función $distr(\eta_1, \eta_2)$ haría ese trabajo.

El Algoritmo 4.1 muestra el procedimiento principal para convertir una fbf a su equivalente CNF. El primer paso conciste en una fase de preprocesamiento para eliminar implicaciones y posteriormente convertir la fbf resultante a su forma normal bajo negación NNF (Algoritmo 4.2). También es nesario distribuir las conjunciones para obtener CNF donde los argumentos de la disyunción son exclusivamente literales (Algoritmo 4.3). El algoritmo para eliminar la implicación IMPL_FREE, se deja como un ejercicio sugerido.

Algoritmo 4.1 CNF

```
1: function CNF(φ)                                   ▷ φ es una fbf proposicional
2:     ψ ← NNF(IMPL_FREE(φ))
3:     switch ψ do
4:         case literal(ψ)
5:             return ψ
6:         case ψ1 ∧ ψ2
7:             return CNF(ψ1) ∧ CNF(ψ2)
8:         case ψ1 ∨ ψ2
9:             return DISTR(CNF(ψ1), CNF(ψ2))
10: end function
```

Algoritmo 4.2 NNF

```
1: function NNF(φ)                                   ▷ φ es libre de implicaciones
2:     switch φ do
3:         case literal(φ)
4:             return φ
5:         case ¬¬φ1
6:             return φ1
7:         case φ1 ∧ φ2
8:             return NNF(φ1) ∧ NNF(φ2)
9:         case φ1 ∨ φ2
10:            return NNF(φ1) ∨ NNF(φ2)
11:        case ¬(φ1 ∧ φ2)
12:            return NNF(¬φ1) ∨ NNF(¬φ2)
13:        case ¬(φ1 ∨ φ2)
14:            return NNF(¬φ1) ∧ NNF(¬φ2)
15: end function
```

Ejemplo 4.10. *Obtenga la CNF de*$(\neg p \wedge q \rightarrow p \wedge (r \rightarrow q))$. *Una vez implementado el procedimiento de eliminación de la implicación, en mi caso en Prolog, podríamos probarlo:*

```
?- impl_free(~p ^ q => p ^ (r => q), IMPLFREE).
IMPLFREE = ( ~ (~p^q)v p^(~r v q))
```

El resultado es correcto, como se demuestra en la Figura 4.6, que además sugiere la estrategia que IMPL_FREE debería seguir. La forma libre de implicación se le puede pasar a NNF:

```
?- impl_free(~p ^ q => p ^ (r => q), IMPLFREE), nnf(IMPLFREE,NNF).
IMPLFREE = ( ~ (~p^q)v p^(~r v q)),
NNF = (( p v ~q)v p^(~r v q))
```

Algoritmo 4.3 DISTR

```
1: function DISTR(η1, η2)                            ▷ η1 y η2 están en CNF
2:     if η1 = η11 ∧ η12 then
3:         return DISTR(η11, η2) ∧ DISTR(η12, η2)
4:     else if η2 = η21 ∧ η22 then
5:         return DISTR(η1, η21) ∧ DISTR(η1, η22)
6:     else                                          ▷ No hay conjunciones
7:         return η1 ∨ η2
8:     end if
9: end function
```

La Figura 4.7 muestra su ejecución.

$$
\begin{aligned}
\texttt{IMPL\_FREE}\,(\phi) &= \neg\texttt{IMPL\_FREE}\,(\neg p \wedge q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \to q)) \\
&= \neg((\texttt{IMPL\_FREE}\,\neg p) \wedge (\texttt{IMPL\_FREE}\,q)) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \to q)) \\
&= \neg((\neg p) \wedge \texttt{IMPL\_FREE}\,q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \to q)) \\
&= \neg(\neg p \wedge q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \to q)) \\
&= \neg(\neg p \wedge q) \vee ((\texttt{IMPL\_FREE}\,p) \wedge \texttt{IMPL\_FREE}\,(r \to q)) \\
&= \neg(\neg p \wedge q) \vee (p \wedge \texttt{IMPL\_FREE}\,(r \to q)) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg(\texttt{IMPL\_FREE}\,r) \vee (\texttt{IMPL\_FREE}\,q))) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee (\texttt{IMPL\_FREE}\,q))) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee q)).
\end{aligned}
$$

Figura 4.6: Eliminación de la implicación con IMPL_FREE

$$
\begin{aligned}
\texttt{NNF}\ (\texttt{IMPL\_FREE}\ \phi) &= \texttt{NNF}\ (\neg(\neg p \wedge q)) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= \texttt{NNF}\ (\neg(\neg p) \vee \neg q) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (\texttt{NNF}\ (\neg\neg p)) \vee (\texttt{NNF}\ (\neg q)) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee (\texttt{NNF}\ (\neg q))) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee ((\texttt{NNF}\ p) \wedge (\texttt{NNF}\ (\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge (\texttt{NNF}\ (\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge ((\texttt{NNF}\ (\neg r)) \vee (\texttt{NNF}\ q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee (\texttt{NNF}\ q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee q)).
\end{aligned}
$$

Figura 4.7: La aplicación de NNF para obtener la forna normal bajo negación.

4.6 EL MUNDO DE TARSKI

Mundo de Tarski

Para evaluar la expresividad de la lógica proposicional usaremos el **Mundo de Tarski** [5]. Se trata de un programa que permite representar medios ambientes bidimensionales poblados de figuras geométricas de diferentes tipos y tamaños. La idea es proponer proposiciones, o conjuntos de ellas, y ver si son falsas o verdaderas en un mundo dado. La ventana principal del programa se muestra en la Figura 4.8.

Como pueden observar, la aplicación tiene dos paneles: El superior, con el título momentáneo de *Untitled World* se usa para crear mundos como los descritos; y el inferior, con el título momentáneo de *Untitled Sentences*, se usa para escribir enunciados en lógica proposicional (aunque como veremos, también lo podemos hacer en primer orden), con ayuda del teclado virtual. Este sugiere que las **proposiciones** que podemos formar se refieren al tipo, tamaño y posición relativa de las figuras en el mundo: *Tet*, *Cube*, *Dodec*, *Small*, *Medium*, *Large*, etc. Por ejemplo, la proposición 1 enuncia que la figura a es un tetraedro y la figura b es un cubo y la figura c es un dodecaedro. Observen que, independientemente de su sintaxis, por ejemplo $Tet(a)$, se trata de proposiciones: "a es un tetraedro".

Proposiciones

El teclado virtual también sugiere los **operadores lógicos** que podemos usar. Por el momento, usaremos la conjunción ($\wedge$), la disyunción ($\vee$), la negación ($\neg$), la implicación ($\to$), la equivalencia ($\leftrightarrow$) y la contradicción ($\bot$). También se sugiere que las figuras en el mundo pueden **etiquetarse** como a, b, c, d, e, f. También conta-

Operadores lógicos

Etiquetas

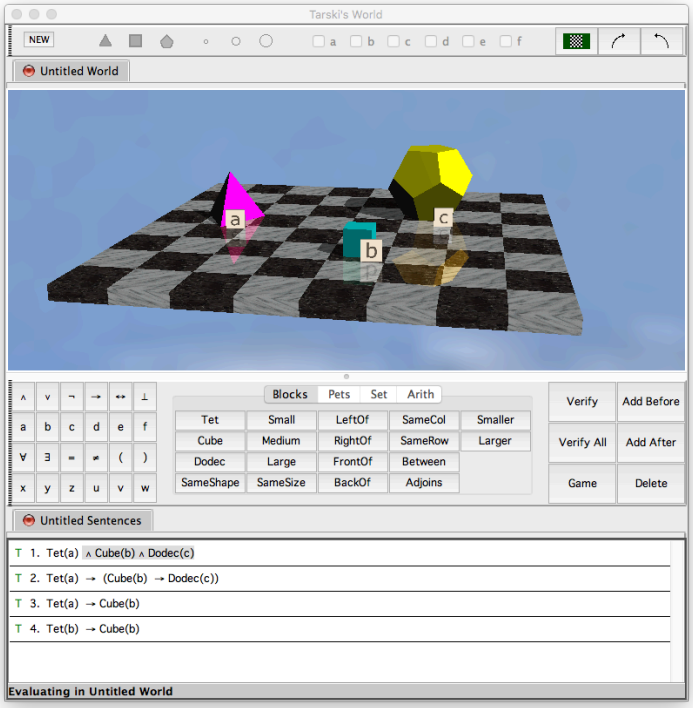

Figura 4.8: Ventana principal del Mundo de Tarski.

mos con paréntesis. El resto de los elementos del teclado, no los usaremos hasta el próximo capítulo.

Tanto el mundo como los enunciados, pueden guardarse en archivos separados para su posterior evaluación. De hecho, la aplicación se instala con una serie de mundos y enunciados que se usarán en los ejercicios propuestos.

4.7 LECTURAS Y EJERCICIOS SUGERIDOS

Este capítulo se basa en las técnicas utilizadas por Huth y Ryan [64] para introducir la lógica proposicional (secciones 1.1 a la 1.4). Como he mencionado, dados los objetivos de este curso; pero sobre todo, dada la duración del mismo, las formas normales de la lógica proposicional y los problemas de satisfacción no han sido abordados aquí. Esta decisión nos permitirá avanzar hacía la lógica de primer orden para abordar temas estrechamente relacionados con estos dos puntos. Para los impacientes, recomiendo las secciones 1.5 y 1.6 del texto en cuestión. Otra referencia al respecto sería el capítulo 7 del libro de Russell y Norvig [107]. Ben-Ari [7] ofrece un texto complementario al de Huth, el material presentado aquí se correspondería con los capítulos 3 y 2 de éste texto. Los capítulos 4,5 y 6 cubren el material que no hemos abordado en nuestro curso. Una presentación más concisa, y por tanto más técnica, de los temas tratados puede encontrarse en el capítulo 1 del libro de Dalen [34]. Por último, estos temas siempre pueden complementarse con la lectura de los fundamentos lógicos de la IA, de Genesereth y Nilsson [50].

Ejercicios sugeridos

Ejercicio 4.1. *Implemente el algoritmo CNF. Para ello es necesario completar las definiciones con el procedimiento para eliminar la impllcación.*