Apuntes de Lógica Matemática

Eduardo González Vaquero
Saúl Rodríguez Martín
Miguel Muñoz Pérez

2 de marzo de 2020

Índice general

1 | Lógica proposicional

1.1. Introducción

Antes de comenzar con definiciones rigurosas, conviene dar una breve explicación sobre la motivación de la lógica proposicional, a la que dedicaremos este capítulo. Un aspecto común de los lenguajes naturales (como el español o el chino) es la ambigüedad de ciertas expresiones, lo cual dificulta las intenciones usuales de la ciencia, a saber, la precisión en la comunicación y la distinción clara de aquello que se comunica. De aquí nace una necesidad evidente, que es cubierta por los lenguajes artificiales y, entre ellos, la lógica formal.

Debemos aclarar entonces cuál es nuestro objeto de estudio. Fijémonos en lo siguiente: existen ciertos elementos del lenguaje que no cambian si los traducimos a otro idioma, por ejemplo. Nos referimos a lo que se denomina como *conectivas lógicas*, es decir, 'no' (negación), 'o' (disyunción), 'y' (conjunción), 'si... entonces' (implicación material) y 'si y solo si' (equivalencia). Obviamente, podemos usar múltiples expresiones distintas para referirnos a cada una de ellas, pero su función en el lenguaje natural no cambia. Por esto mismo podemos formalizarlas y estudiarlas desde el punto de vista matemático.

Ahora bien, a fin de sistematizar nuestro lenguaje, introducimos una serie de símbolos que se referirán a las entidades correspondientes del lenguaje natural. Conviene puntualizar de nuevo que estas traducciones, es decir, estas 'asignaciones de significados' *no* son únicas. Esto lo veremos más adelante. Entonces tenemos: $\neg$ (negación), $\vee$ (disyunción), $\wedge$ (conjunción), $\rightarrow$ (implicación material) y $\leftrightarrow$ (equivalencia).

Los otros elementos básicos que constituirán nuestro lenguaje formal son las *proposiciones*. Con el fin de no entrar en cuestiones semánticas que exceden nuestro estudio, establecemos la siguiente:

Definición 1.1. Una proposición es un enunciado que puede ser verdadero o falso.

Como con las conectivas, introducimos una serie de *símbolos de proposición* para referirnos formalmente a las proposiciones. Denotamos por SP al conjunto

de todos estos símbolos.

Así, el enunciado 'Todos los solteros son hombres no casados' es una proposición, mientras que '¿Va a llover mañana?' no lo es. Por otro lado, 'César conquistó las Galias y combatió contra Vercingétorix' admite una descomposición como conjunción de dos proposiciones. Evidentemente, esta reducción a proposiciones más simples tiene que parar eventualmente; así llegamos a lo que denominamos *proposiciones atómicas*. Esto nos permitirá, una vez determinada la verdad o falsedad de estas proposiciones atómicas, hallar la de todas las proposiciones que se compongan de ellas mediante la aplicación de las conectivas lógicas (antiguamente llamadas *proposiciones moleculares*).

Las proposiciones atómicas se representan entonces por los elementos de SP. Añadimos, por motivos que serán aclarados más adelante, dos elementos $\top$ y $\bot$, que se refieren a la *proposición idénticamente verdadera* y a la *proposición idénticamente falsa*, respectivamente[1]. Es decir,

Definición 1.2. Definimos el conjunto de las proposiciones atómicas como:

$$AT := SP \cup \{\top, \bot\}$$

Definición 1.3. (Alfabeto) Dado un conjunto de símbolos de proposiciones SP, se denomina alfabeto a:

$$A_{SP} = SP \cup \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,), \top, \bot\},$$

obtenido añadiendo a SP los símbolos $\top$ y $\bot$, las conectivas lógicas $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, y los paréntesis, $(,)$.

Definición 1.4. (Cierre de Kleene) Dado un conjunto de símbolos A, definimos el cierre de Kleene de A, A^*, como el conjunto de las concatenaciones de elementos de A junto con ϵ, el *espacio vacío* o en blanco.

Ejemplo 1.5. Dado $A = \{a, b\}$, le corresponde

$$A^* = \{\epsilon, a, b, ab, ba, aab, baa, aba, ...\}$$

Ejemplo 1.6. A^*_{SP} será el conjunto de todas las concatenaciones de elementos de A_{SP}, junto con ϵ.

Una vez que disponemos de los elementos básicos de la sintaxis del lenguaje y sus sucesivas combinaciones, nos interesa definir las expresiones que están 'bien formadas', es decir, queremos excluir de nuestro lenguaje concatenaciones de símbolos como '$)) \leftrightarrow \varphi$'.

Definición 1.7. (Proposiciones posibles) Se define el conjunto de proposiciones posibles, $PROP_{SP}$, como la intersección de todos los subconjuntos de A^*_{SP} que verifican:

[1] En realidad, como veremos, al igual que $\neg, \wedge, \vee, \rightarrow$ y $\leftrightarrow$ son conectivas binarias, $\top$ y $\bot$ pueden considerarse conectivas 0-arias.

1. $SP \subseteq PROP_{SP}$.

2. Si $\phi \in PROP_{SP}$, entonces $(\neg\phi) \in PROP_{SP}$.

3. Si $\phi_1, \phi_2 \in PROP_{SP}$ y $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, entonces $(\phi_1 \square\ \phi_2) \in PROP_{SP}$.

Es fácil comprobar que $PROP_{SP}$ también cumple estas tres propiedades, por tanto decimos que es el mínimo subconjunto que las verifica. También podemos obtener $PROP_{SP}$ de forma constructiva:

Proposición 1.8. *Se define $P_0 := AT$ y para cada $n \in \mathbb{N}$:*

$$P_n := P_{n-1} \cup \{(\neg\phi) : \phi \in P_{n-1}\} \cup \{(\phi_1 \square\ \phi_2) : \phi_1, \phi_2 \in P_{n-1}, \square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}\}.$$

Entonces $P := \bigcup_{n=0}^{\infty} P_n$ coincide con $PROP_{SP}$.

Demostración. Comenzamos comprobando $P \subseteq PROP_{SP}$. Para ello, comprobamos por inducción que $P_n \subseteq PROP_{SP}$ para todo $n \in \mathbb{N}$.

Está claro que $P_0 \subseteq PROP_{SP}$, ya que $PROP_{SP}$ cumple la propiedad 1. Ahora, suponiendo que $P_n \subseteq PROP_{SP}$, por las propiedades 2 y 3, $\{(\neg\phi) : \phi \in P_{n-1}\} \cup \{(\phi_1 \square\ \phi_2) : \phi_1, \phi_2 \in P_{n-1}, \square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}\}$ también estarán contenidos en $PROP_{SP}$. Por tanto, $P_{n+1} \subseteq PROP_{SP}$, y hemos acabado la inducción.

Veamos ahora que $PROP_{SP} \subseteq P$. Para ello nos basta comprobar que P cumple las tres propiedades de la definición 1.7. La propiedad 1 es directa, pasamos a comprobar las propiedades 2 y 3.

Dado $\varphi \in P$, existe $n \in \mathbb{N} \cup \{0\}$ tal que $\varphi \in P_n$ luego $(\neg\varphi) \in P_{n+1}$ por lo que $(\neg\varphi) \in P$.

Sean $\varphi_1, \varphi_2 \in P$ y sea $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. Tomamos $n \in \mathbb{N}$ tal que $\varphi_1, \varphi_2 \in P_n$ (para cada uno existe al menos un natural: basta tomar el mayor). Luego $(\phi_1 \square\ \phi_2) \in P_{n+1}$ y por lo tanto también pertenece a P.

□

A partir de ahora, usaremos para mayor brevedad el símbolo $\square$ como intercambiable por $\wedge, \vee, \rightarrow$ o $\leftrightarrow$ a no ser que se indique lo contrario.

Básicamente, el resultado anterior nos dice que podemos construir cualquier proposición partiendo de proposiciones atómicas y creando proposiciones más complicadas a partir de ellas usando las conectivas. Esta forma de verlo da lugar al siguiente concepto:

Definición 1.9. (Secuencia de formación) Definimos una *secuencia de formación* para cierto $\varphi \in A^*_{SP}$ como una sucesión finita $\langle \varphi_1, \ldots, \varphi_n \rangle$ de elementos de A^*_{SP} tal que $\varphi_n = \varphi$ y para cada $i \leq n$, se da alguna de estas tres opciones:

- $\varphi_i \in AT$.
- $\varphi_i = (\neg\varphi_j)$, para cierto $j < i$.
- $\varphi_i = (\varphi_j \square \varphi_k)$, para ciertos $j, k < i$.

Como veremos en los ejercicios, toda proposición puede construirse mediante una secuencia de formación, y recíprocamente si φ tiene una secuencia de formación, entonces $\varphi \in PROP_{SP}$.

1.2. Inducción estructural y recursión

Análogamente al método de inducción en los naturales, el método de inducción estructural nos servirá para demostrar que todas las fórmulas cumplen una *propiedad*[2] en dos pasos: demostrar que todas las proposiciones atómicas cumplen dicha propiedad, y demostrar que si dos proposiciones φ_1, φ_2 cumplen una propiedad, entonces $(\neg\varphi_1)$ y $(\varphi_1 \square \varphi_2)$ también la cumplen.

Proposición 1.10. *(Inducción estructural) Supongamos que queremos probar $P(\varphi)$[3] para todo $\varphi \in PROP_{SP}$ (siendo P una propiedad cualquiera). Entonces basta comprobar que:*

1. *$P(\varphi)$ para toda $\varphi \in AT$.*
2. *Si $P(\varphi)$ entonces $P((\neg\varphi))$.*
3. *Si $P(\varphi_1)$ y $P(\varphi_2)$ entonces $P((\varphi_1 \square \varphi_2))$.*

Demostración. Si una propiedad P cumple 1, 2 y 3, entonces el conjunto de proposiciones que cumplen P cumple las tres propiedades de 1.7, por tanto contiene a $PROP_{SP}$.

□

Ejemplo 1.11. Definamos $P(\varphi)$ como la propiedad de $|\varphi|_) = |\varphi|_($, siendo $|\varphi|_)$ el número de símbolos) en la fórmula φ y $|\varphi|_($ el número de ([4]. Vamos a comprobar por inducción estructural que todas las proposiciones cumplen esta propiedad:

1. Para cualquier $\varphi \in AT$ se verifica trivialmente $|\varphi|_) = |\varphi|_(= 0$, luego $P(\varphi)$.
2. Supongamos $P(\varphi)$ con $\varphi \in PROP_{SP}$, entonces $|\varphi|_(= |\varphi|_)$ y por tanto:

$$|(\neg\varphi)|_(= |\varphi|_(+ 1 = |\varphi|_) + 1 = |(\neg\varphi)|_)$$

luego $P((\neg\varphi))$.

[2] No hemos definido rigurosamente el concepto de "propiedad", pero esencialmente significa algo que las proposiciones pueden cumplir o no. Una forma rigurosa de definir propiedad consiste en definirla como un subconjunto $P \subseteq PROP_{SP}$, de modo que decimos que una proposición cumple la propiedad P si y solo si pertenece a P.

[3] Usamos la notación $P(\varphi)$ para decir que la fórmula φ cumple la propiedad p.

[4] En general, siendo $*$ un símbolo, podemos denotar $|\varphi|_*$ al número de símbolos $*$ en la fórmula φ.

3. Supongamos $P(\varphi), P(\psi)$ con $\varphi, \psi \in PROP_{SP}$, entonces:

$$|(\varphi\square\psi)|_(= |\varphi|_(+ |\psi|_(+ 1 = |\varphi|_) + |\psi|_) + 1 = |(\varphi\square\psi)|_)$$

luego $P((\varphi\square\psi))$.

Definición 1.12. (Prefijos) Sea A un alfabeto y sea $w \in A^*$. Decimos que w' es prefijo de w si existe w'' tal que $w = w'w''$, es decir, si $w = w_1w_2...w_n$ entonces existe $0 \leq k \leq n$ tal que $w' = w_1...w_k$.
Decimos además que un prefijo w' es prefijo propio de w si es distinto de w y ϵ.

Ejemplo 1.13. Sea $A := \{a, b\}$ y sea $w := aababb$. Entonces sus prefijos son $\epsilon, a, aa, aab, aaba, aabab, aababb$, correspondientes a $k = 0, 1, 2, 3, 4, 5, 6$, respectivamente, y sus prefijos propios son $a, aa, aab, aaba, aabab$.

Proposición 1.14. *Si φ' es prefijo propio de φ, entonces $|\varphi'|_) < |\varphi'|_($.*

Demostración. Se demuestra por inducción estructural.

1. Si $\varphi \in AT$ no tiene prefijos propios luego se cumple la afirmación.
2. Sea $\varphi \in PROP_{SP}$ cumpliendo la propiedad del enunciado y sea φ' un prefijo propio de $(\neg\varphi)$. Existen cuatro casos:
   a) $\varphi' = ($, entonces $|\varphi'|_(= 1 > 0 = |\varphi'|_)$-
   b) $\varphi' = (\neg$, ocurre lo mismo que antes.
   c) $\varphi' = (\neg\varphi''$ con φ'' prefijo propio de φ entonces por inducción se verifica $|\varphi''|_) < |\varphi''|_($. Ahora bien, $|\varphi'|_(= |\varphi''|_(+ 1$ y $|\varphi'|_) = |\varphi''|_)$ luego:
   $$|\varphi'|_) < |\varphi'|_($$
   d) $\varphi' = (\neg\varphi$, que es directo.
3. Se hace de forma similar al apartado anterior. Basta probar con los prefijos $($, $(\varphi_1'$, $(\varphi_1$, $(\varphi_1\square$, $(\varphi_1\square\varphi_2'$ y $(\varphi_1\square\varphi_2$.

□

En próximas secciones será útil, dado un conjunto A, definir una función $H : PROP_{SP} \to A$ conociendo su restricción a AT, y suponiendo que podemos calcular $H((\neg\varphi))$ y $H((\varphi\square\psi))$ en función de $H(\varphi)$ y $H(\psi)$. Si la función H queda bien definida así, diremos que está *definida de forma recursiva*. La siguiente proposición nos garantiza que podemos definir funciones de esta forma:

Proposición 1.15. *El esquema de definición recursiva da como resultado una única función, es decir, dadas:*

1. $H_{AT} : AT \to A$.
2. $H_\neg : A \to A$.

3. $H_{\square} : A \times A \to A$.

existe una única función $H : PROP_{SP} \to A$ *tal que:*

1. $H(\varphi) = H_{AT}(\varphi)$, *para toda* $\varphi \in AT$.
2. $H((\neg\varphi)) = H_{\neg}(H(\varphi))$.
3. $H((\varphi_1 \square \varphi_2)) = H_{\square}(H(\varphi_1), H(\varphi_2))$, *para cada* $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$.

Demostración. Se omite la demostración. □

Ejemplo 1.16. Sea $H : PROP_{SP} \to \mathbb{N}$ la función $|.|_{(}$ definida anteriormente. Podemos definirla de manera recursiva:

1. $H(\varphi) = 0$, para toda $\varphi \in AT$.
2. $H((\neg\varphi)) = H(\varphi) + 1$.
3. $H((\varphi_1 \square \varphi_2)) = H(\varphi_1) + H(\varphi_2) + 1$.

Nótese como en ocasiones omitimos las funciones auxiliares y usamos la notación H para referirnos a H, H_{AT}, $H_{\neg}$ y $H_{\square}$. Este abuso de notación no suele dar lugar a confusión.

Ejemplo 1.17. Podemos definir recursivamente una función que nos lleve cada fórmula al conjunto de todas sus subfórmulas, es decir, $SUB : PROP_{SP} \to \mathcal{P}(PROP_{SP})$[5] tal que:

1. $SUB(\varphi) = \{\varphi\}$, para toda $\varphi \in AT$.
2. $SUB((\neg\varphi)) = SUB(\varphi) \cup \{(\neg\varphi)\}$.
3. $SUB((\varphi_1 \square \varphi_2)) = SUB(\varphi_1) \cup SUB(\varphi_2) \cup \{(\varphi_1 \square \varphi_2)\}$.

1.3. Eliminación de paréntesis

Para conseguir una notación más compacta, podemos establecer una notación que use menos paréntesis cuando nos convenga. Sin embargo, la omisión de paréntesis puede dar lugar a ambigüedades en las fórmulas, por ejemplo,

$$p \to q \to r$$

podría simbolizar dos fórmulas distintas:

- $((p \to q) \to r)$
- $(p \to (q \to r))$

[5] Dado un conjunto X, denotamos por $\mathcal{P}(X)$ a su potencia, es decir, el conjunto de subconjuntos de X.

Para evitar este tipo de situaciones, necesitamos las siguientes reglas de omisión de paréntesis:

1. Los paréntesis externos pueden omitirse. Esto no da lugar a ambigüedad.
2. Las conectivas se aplicarán en este orden: $\neg, \wedge, \vee, \to, \leftrightarrow$.
3. Cuando hay varias conectivas del mismo tipo seguidas, se asocia siempre por la izquierda.

Algunos ejemplos de aplicación de estas reglas:

$$\begin{array}{ccc} \neg p \vee p & \text{es} & ((\neg p) \vee p) \\ \neg p \wedge q \vee s \to t \leftrightarrow r & \text{es} & (((((\neg p) \wedge q) \vee s) \to t) \leftrightarrow r) \\ p \to q \to s & \text{es} & ((p \to q) \to s) \\ p \wedge q \leftrightarrow \neg s \to t \vee r & \text{es} & ((p \wedge q) \leftrightarrow ((\neg s) \to (t \vee r))) \\ t \leftrightarrow p \to \neg q \vee p \to s & \text{es} & (t \leftrightarrow ((p \to ((\neg q) \vee p)) \to s)) \end{array}$$

Comprobaremos que las conectivas $\wedge$ y $\vee$ son asociativas y conmutativas, por tanto, en ese sentido, no importa el orden en que se asocien. De modo que, cuando una conectiva se repita varias veces seguidas, podemos usar la siguiente notación:

$$\bigwedge_{i \in I} p_i := p_{i_1} \wedge \cdots \wedge p_{i_n}$$

$$\bigvee_{i \in I} p_i := p_{i_1} \vee \cdots \vee p_{i_n},$$

siendo $I = \{i_1, \ldots, i_n\}$ un conjunto finito de índices.
Establecemos además el siguiente convenio,

$$\bigwedge_{i \in \emptyset} p_i := \top$$

$$\bigvee_{i \in \emptyset} p_i := \bot,$$

que será consistente con el comportamiento semántico de $\vee$ y $\wedge$.

1.4. Asignaciones de verdad

Hasta ahora hemos venido considerando cuestiones sintácticas sobre las fórmulas. Lo que queremos ahora es abordar su semántica, esto es, su comportamiento cuando les damos una interpretación determinada. Nos interesa, por tanto, estudiar qué significa que una proposición se siga de otra independientemente de la traducción que hagamos al lenguaje natural, esto es, la noción de *consecuencia lógica*. Así, por ejemplo, queremos que q se siga de $p \wedge (p \to q)$ independientemente de cómo se traduzcan p y q informalmente. Es decir, exigimos que, sea cual sea la traducción de $p \wedge (p \to q)$, si es verdadera entonces la de q tiene que serlo también. Siguiendo la definición 1.1,

Definición 1.18. Definimos el conjunto de valores de verdad, $Bool = \{V, F\}$, donde V es *la verdad* y F es *la falsedad*[6].

Una vez que disponemos de $Bool$ podemos definir la aplicación que nos lleva cada proposición a su valor de verdad correspondiente. Vamos a definir esta aplicación de forma recursiva, es decir, primero lo haremos para las fórmulas atómicas y después describiremos reglas para deducirla en el resto de fórmulas.

Definición 1.19. (Valoración) Sea el conjunto $Bool = \{V, F\}$. Una valoración o evaluación es una función $v : SP \to Bool$.

Un alfabeto de n letras tiene 2^n valoraciones distintas.

Para definir la valoración de proposiciones, vamos a necesitar una forma de interpretar las conectivas. Con este fin, a cada conectiva le asociaremos una aplicación, que definiremos caso a caso en formato de tabla de verdad:

1. $v_\neg : Bool \to Bool$;

φ	$v_\neg(\varphi)$
V	F
F	V

2. $v_\wedge, v_\vee, v_\to, v_\leftrightarrow : Bool \times Bool \to Bool$;

φ_1	φ_2	$v_\wedge(\varphi_1, \varphi_2)$	$v_\vee(\varphi_1, \varphi_2)$	$v_\to(\varphi_1, \varphi_2)$	$v_\leftrightarrow(\varphi_1, \varphi_2)$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	V	V	F
F	F	F	F	V	V

Decimos que la conectiva binaria $\wedge$ es un operador conmutativo ya que $v_\wedge(V, F) = v_\wedge(F, V)$. $\vee$ y $\leftrightarrow$ también son conmutativas.

Definición 1.20. (Extensión) Dada la valoración $v : SP \to Bool$, definimos recursivamente su *extensión*, $\hat{v} : PROP_{SP} \to Bool$:

1. $\hat{v}(\bot) = F$.
2. $\hat{v}(\top) = V$.[7]

[6] Nada nos impide considerar casos en que $Bool$ sea infinito numerable, o incluso no numerable, pero la lógica concebida clásicamente es bivalente, esto es, solo consta de dos valores de verdad.

[7] Aquí es donde hemos dado sentido a la definición que dimos de $\top$ y $\bot$ en 1.2.

3. $\hat{v}(\varphi) = v(\varphi)$, para toda $\varphi \in SP$.

4. $\hat{v}((\neg\varphi)) = v_{\neg}(\hat{v}(\varphi))$.

5. $\hat{v}((\varphi_1 \square \varphi_2)) = v_{\square}(\hat{v}(\varphi_1), \hat{v}(\varphi_2))$.

La propia definición nos da un método para calcular los valores de verdad de fórmulas en función de los de sus símbolos:

Ejemplo 1.21. Encontremos el valor de verdad de $p \wedge \neg q \to r \vee s$, siendo v una valoración que cumple $v(p) = v(q) = V, v(r) = v(s) = F$:

1. $\hat{v}(\neg q) = v_{\neg}(\hat{v}(q)) = v_{\neg}(V) = F$.

2. $\hat{v}(p \wedge \neg q) = v_{\wedge}(\hat{v}(p), \hat{v}(\neg q)) = v_{\wedge}(V, F) = F$.

3. $\hat{v}(r \vee s) = v_{\vee}(\hat{v}(r), \hat{v}(s)) = v_{\vee}(F, F) = F$.

4. Finalmente, $\hat{v}(p \wedge \neg q \to r \vee s) = v_{\to}(\hat{v}(p \wedge \neg q), \hat{v}(r \vee s)) = v_{\to}(F, F) = V$.

Definición 1.22. (satisfactibilidad) Dada $v : SP \to Bool$ y $\varphi \in PROP_{SP}$ se dice que v satisface φ, y se escribe como $v \vDash \varphi$, si y solo si $\hat{v}(\varphi) = V$. Si $\hat{v}(\varphi) = F$, escribimos $v \nvDash \varphi$.

En virtud de la definición anterior, podemos clasificar cada fórmula φ como:

- *Satisfactible*, si existe alguna valoración v tal que $v \vDash \varphi$.
  - *Tautología*, si $v \vDash \varphi$ para toda valoración v.
  - *Contingencia*, si es satisfactible pero no tautología.
- *Contradicción*, si no es satisfactible.

Ejemplo 1.23. La fórmula $p \wedge q \to r \leftrightarrow (p \to q) \to r$ es contingencia. Invitamos al lector a que realice la tabla de verdad correspondiente.

Definición 1.24. (Conjunto de modelos) Dado $\Phi \subseteq PROP_{SP}$, el conjunto de modelos de Φ se define como:

$$Mod(\Phi) := \{v : SP \to Bool \mid \text{para toda } \varphi \in \Phi \ v \vDash \varphi\}$$

Definición 1.25. (Satisfactibilidad) Sea $\Phi \subseteq PROP_{SP}$.

Φ se dice satisfactible, y se escribe $Sat(\Phi)$, si $Mod(\Phi) \neq \emptyset$.

Φ se dice insatisfactible, y se escribe $Insat(\Phi)$, si $Mod(\Phi) = \emptyset$.

Ahora llegamos a la definición del concepto que ha motivado nuestro estudio inicial de la semántica.

Definición 1.26. (Consecuencia lógica) Sea $\varphi \in PROP_{SP}$. Decimos que φ es consecuencia lógica de Φ, $\Phi \vDash \varphi$, si y solo si $Mod(\Phi) \subseteq Mod(\{\varphi\})$.

Esto equivale a decir que toda asignación de verdad que satisface todos los elementos de Φ también satisface φ.

De ahora en adelante, escribimos $Mod(\varphi)$ en vez de $Mod(\{\varphi\})$.

Ejemplo 1.27. Dado Φ tal que $Insat(\Phi)$, se sigue de la anterior definición que $\Phi \vDash \varphi$, para cualquier φ.

A continuación presentamos algunos resultados básicos de semántica.

Proposición 1.28.

1. *$Mod(\emptyset) = \{v | v : SP \to Bool\}$.*
2. *Si $\Phi = \emptyset$ y $\Phi \vDash \varphi$, entonces φ es tautología.*
3. *Demostrar que $Mod(PROP_{SP}) = \emptyset$.*
4. *Demostrar que $Mod(\Phi) \cap Mod(\Psi) = Mod(\Phi \cup \Psi)$.*
5. *Demostrar que $Mod(\Phi) \cup Mod(\Psi) \subseteq Mod(\Phi \cap \Psi)$.*

Demostración.

1. En efecto, toda valoración satisface todas las proposiciones del conjunto vacío.
2. $Mod(\emptyset)$ tiene a todas las valoraciones, por tanto si $Mod(\emptyset) \subseteq Mod(\varphi)$, cualquier valoración satisface φ.
3. Dada una valoración v, y cualquier proposición φ, no podemos tener a la vez $\hat{v}(\varphi) = T$ y $\hat{v}(\neg\varphi) = T$. Por tanto v no satisface todos los elementos de $PROP_{SP}$.
4. Comprobamos los dos contenidos:
   $\subseteq$: Si una valoración v está en $(\Phi) \cap Mod(\Psi)$, satisface todos los elementos de Φ y de Ψ, por tanto satisface todos los elementos de $\Phi \cup \Psi$.
   $\supseteq$: Si v está en $Mod(\Phi \cup \Psi)$, v satisface todos los elementos de $\Phi \cup \Psi$, por tanto satisface todos los elementos de Φ y todos los de Ψ, es decir, $v \in Mod(\Phi) \cap Mod(\Psi)$.
5. Dada $v \in Mod(\Phi) \cup Mod(\Psi)$, o bien v verifica todos los elementos de Φ o los de Ψ. En ambos casos, v verifica los elementos de $\Phi \cap \Psi$.

En 5, el contenido opuesto no se cumple. Poniendo por ejemplo $\Phi = \{p\}, \Psi = \{p \wedge p\}$, siendo $p \in SP$, tenemos $\Phi \cap \Psi = \emptyset$, por tanto $Mod(\Phi \cap \Psi)$ contiene a todas las valoraciones, pero cualquier valoración v que cumpla $v(p) = F$ no estará en $Mod(\Phi) \cup Mod(\Psi)$. □

Proposición 1.29. *$\Phi \cup \{\varphi\} \vDash \psi$ si y solo si $\Phi \vDash \varphi \to \psi$.*

Demostración. Veamos que $Mod(\Phi) \subseteq Mod(\varphi \to \psi)$ suponiendo que $Mod(\Phi \cup \{\varphi\}) \subseteq Mod(\psi)$. Sea $v \in Mod(\Phi)$. Si $\hat{v}(\varphi) = V$ entonces $v \in Mod(\varphi)$. Por hipótesis, $v \in Mod(\psi)$, luego $\hat{v}(\psi) = V$ y entonces $\hat{v}(\varphi \to \psi) = V$, es decir, $v \in Mod(\varphi \to \psi)$. Si $\hat{v}(\varphi) = F$, $\hat{v}(\varphi \to \psi) = V$ y de nuevo $v \in Mod(\varphi \to \psi)$.

Recíprocamente, si $v \in Mod(\Phi \cup \{\varphi\})$, entonces $\hat{v}(\varphi) = V$. Por hipótesis, al ser $v \in Mod(\Phi)$, $v \in Mod(\varphi \to \psi)$, es decir, $\hat{v}(\varphi \to \psi) = V$, luego $\hat{v}(\psi) = V$ y $v \in Mod(\psi)$. □

Proposición 1.30. *$\Phi \vDash \varphi$ si y solo si $Insat(\Phi \cup \{\neg\varphi\})$*

Demostración. Supongamos que $\Phi \vDash \varphi$, esto es, que $Mod(\Phi) \subset Mod(\varphi)$. Sea $v \in Mod(\Phi \cup \{\neg\varphi\})$. Entonces, por definición, $v \vDash \chi$, para toda $\chi \in \Phi \cup \{\neg\varphi\}$. Luego en especial, $v \vDash \neg\varphi$, lo que contradice que $v \in Mod(\varphi)$ (¿por qué?).

Recíprocamente, si $Mod(\Phi \cup \{\neg\varphi\}) = \emptyset$, sea $v \in Mod(\Phi)$. Entonces $v \notin Mod(\neg\varphi)$, por tanto $\hat{v}(\neg\varphi) = F$, pero como $\hat{v}(\neg\varphi) = v_\neg(\hat{v}(\varphi))$, tenemos que $F = v_\neg(\hat{v}(\varphi))$, es decir, $\hat{v}(\varphi) = V$, así que $v \in Mod(\varphi)$. Como esto lo cumple cualquier $v \in Mod(\Phi)$, tenemos, como queríamos, que $Mod(\Phi) \subseteq Mod(\varphi)$

□

De ahora en adelante, escribiremos $\Phi, \varphi \vDash \psi$ en vez de $\Phi \cup \{\varphi\} \vDash \psi$.

Dada una valoración v, para cualesquiera fórmulas φ_1, φ_2 en que aparecen los símbolos $p_1, \ldots, p_n$ y $q_1, \ldots, q_m$, respectivamente, está claro, por cómo hemos definido las valoraciones, que $\hat{v}(\varphi_1)$ depende de $v(p_1), \ldots, v(p_n)$, y $\hat{v}(\varphi_2)$ depende de $v(q_1), \ldots, v(q_m)$. De modo que para comprobar que $\varphi_1 \vDash \varphi_2$ nos basta con comprobar que, para cada una de las posibles valoraciones de $p_1, \ldots, p_n, q_1, \ldots, q_m$, $\hat{v}(\varphi_2) = V$ si $\hat{v}(\varphi_1) = V$. Este es nuestro primer procedimiento para demostrar consecuencias lógicas entre fórmulas, que resumiremos gráficamente en tablas de verdad.

Ejemplo 1.31. Consideremos $\Phi = \{\neg p \wedge q \to r,\ \neg p,\ \neg r\}$, $\varphi = \neg q$. Para demostrar que $\Phi \vDash \varphi$ basta realizar la tabla de verdad correspondiente y comparar los valores de verdad de las premisas del conjunto Φ con los de φ. En Φ y φ aparecen los símbolos p, q y r, por tanto tenemos la siguiente tabla de verdad:

p	q	r	$(\neg p \wedge q) \to r$	$\neg p$	$\neg r$	$\neg q$
V	V	V	V	F	F	F
V	V	F	V	F	V	F
V	F	V	V	F	F	V
V	F	F	V	F	V	V
F	V	V	V	V	F	F
F	V	F	F	V	V	F
F	F	V	V	V	F	V
F	F	F	V	V	V	V

En efecto, solo hay un caso en que se cumplen todas las proposiciones de Φ (la última fila) y en ese caso se cumple φ. Por tanto, $\Phi \vDash \varphi$.

1.5. Equivalencia lógica

El método descrito de las tablas de verdad aumenta de complejidad a medida que lo hace el número de proposiciones que estudiamos. Por ello, investigamos relaciones formales entre las fórmulas que simplifiquen tales cuestiones. Comenzamos por las proposiciones atómicas:

Definición 1.32. (Equivalencia lógica) Sean $\varphi, \psi \in PROP_{SP}$. φ se dice lógicamente equivalente a ψ, $\varphi \sim \psi$, si para toda valoración v, $\hat{v}(\varphi) = \hat{v}(\psi)$.

Esto equivale a decir que $\varphi_1 \vDash \varphi_2$ y $\varphi_2 \vDash \varphi_1$.

Proposición 1.33. *$\sim$ es relación de equivalencia. Además, $\sim$ es congruencia respecto de las conectivas lógicas, es decir:*

1. *Si $\varphi \sim \psi$, entonces $\neg\varphi \sim \neg\psi$.*
2. *Si $\varphi_1 \sim \varphi_2$ y $\psi_1 \sim \psi_2$, entonces $\varphi_1 \square \psi_1 \sim \varphi_2 \square \psi_2$.*

Demostración. Que es relación de equivalencia es directo.

Supongamos $\varphi \sim \psi$, y sea v cualquier valoración. Entonces

$$\hat{v}(\neg\varphi) = v_\neg(\hat{v}(\varphi)) = v_\neg(\hat{v}(\psi)) = \hat{v}(\neg\psi).$$

De igual manera, supongamos $\varphi_1 \sim \varphi_2$ y $\psi_1 \sim \psi_2$, sea v cualquier valoración. Entonces

$$\hat{v}(\varphi_1 \square \psi_1) = v_\square(\hat{v}(\varphi_1), \hat{v}(\psi_1)) = v_\square(\hat{v}(\varphi_2), \hat{v}(\psi_2)) = \hat{v}(\varphi_2 \square \psi_2).$$

□

Las siguientes reglas nos ayudarán a demostrar equivalencias entre fórmulas:

Teorema 1.34. *(Leyes de equivalencia lógica)*

1. *Conmutatividad:*

$$p \wedge q \sim q \wedge p, \quad p \vee q \sim q \vee p, \quad p \leftrightarrow q \sim q \leftrightarrow p$$

2. *Asociatividad:*

$$(p \vee q) \vee r \sim p \vee (q \vee r), \quad (p \wedge q) \wedge r \sim p \wedge (q \wedge r)$$

3. *Idempotencia:*

$$p \wedge p \sim p, \quad p \vee p \sim p$$

4. *Distributiva:*

$$p \wedge (q \vee r) \sim (p \wedge q) \vee (p \wedge r), \quad p \vee (q \wedge r) \sim (p \vee q) \wedge (p \vee r)$$

5. *Absorción:*

$$p \wedge (p \vee q) \sim p, \quad p \vee (p \wedge q) \sim p$$

6. *Cero y unidad:*

$$p \wedge \top \sim p, \quad p \vee \top \sim \top, \quad p \wedge \bot \sim \bot, \quad p \vee \bot \sim p$$

7. *Contradicción:*

$$p \wedge \neg p \sim \bot$$

8. *Tercio excluso:*

$$p \vee \neg p \sim \top$$

9. *Negación:*

$$\neg\neg p \sim p$$

10. *Leyes de De Morgan:*

$$\neg(p \wedge q) \sim \neg p \vee \neg q, \quad \neg(p \vee q) \sim \neg p \wedge \neg q$$

11. *Simplificación de los condicionales:*

$$p \to q \sim \neg p \vee q, \quad p \leftrightarrow q \sim (p \to q) \wedge (q \to p)$$

Demostración. Las demostraciones de equivalencia consisten simplemente en comprobar que coinciden las tablas de verdad. Comprobamos las tres primeras:

1. Conmutatividad:

p	q	$p \wedge q$	$q \wedge p$	$p \vee q$	$q \vee p$	$p \leftrightarrow q$	$q \leftrightarrow p$
V	V	V	V	V	V	V	V
V	F	F	F	V	V	F	F
F	V	F	F	V	V	F	F
F	F	F	F	F	F	V	V

2. Asociatividad:

p	q	r	$(p \vee q) \vee r$	$p \vee (q \vee r)$	$(p \wedge q) \wedge r$	$p \wedge (q \wedge r)$
V	V	V	V	V	V	V
V	V	F	V	V	F	F
V	F	V	V	V	F	F
V	F	F	V	V	F	F
F	V	V	V	V	F	F
F	V	F	V	V	F	F
F	F	V	V	V	F	F
F	F	F	F	F	F	F

3. Idempotencia:

p	$p \wedge p$	$p \vee p$
V	V	V
F	F	F

□

1.6. Sustitución

Pasamos a estudiar los lemas de sustitución, que entre otras cosas nos permitirán generalizar las leyes de equivalencia lógica. Para ello es necesario estudiar el concepto de *sustitución*:

Definición 1.35. (Sustitución) Sean $\varphi, \psi \in PROP_{SP}$ y sea $p \in SP$. Llamamos $\psi[\varphi/p]$ a la fórmula que resulta de sustituir en ψ cada aparición de p por φ. Dados φ y p, podemos definir $\psi[\varphi/p]$ recursivamente del siguiente modo:

1. $p[\varphi/p] = \varphi$.
2. $q[\varphi/p] = q$, para cualquier $q \in SP \setminus \{p\}$.

3. $\top[\varphi/p] = \top$.

4. $\bot[\varphi/p] = \bot$.

5. $(\neg\psi)[\varphi/p] = \neg(\psi[\varphi/p])$.

6. $(\psi_1 \Box \psi_2)[\varphi/p] = (\psi_1[\varphi/p] \Box \psi_2[\varphi/p])$.

Ejemplo 1.36. Si usamos las fórmulas $\psi = p \wedge q \to p$ y $\varphi = p_1 \vee \neg p_2$, al sustituir p por φ en ψ obtenemos la fórmula:

$$\psi[\varphi/p] = (p_1 \vee \neg p_2) \wedge q \to (p_1 \vee \neg p_2).$$

Introducimos la siguiente notación que será de utilidad: dada una función $f : A \to B$ y dados $a \in A$, $b \in B$, denotamos respectivamente por $f[b/a](x)$ a $f(x)$, si $x \neq a$, y a b, si $x = a$.

Ejemplo 1.37. Sea $X \in \{V, F\}$. Si con la notación anterior usamos como función una valoración v, $v[X/p]$ es una valoración de verdad que cumple:

- $v[X/p](q) = v(q)$ si $q \neq p$
- $v[X/p](p) = X$.

La siguiente proposición muestra que obtenemos el mismo resultado tanto si sustituimos y posteriormente asignamos un valor de verdad como si hacemos el proceso inverso:

Proposición 1.38. *Sean $\varphi, \psi \in PROP_{SP}$, $p \in SP$ y $v : SP \to Bool$ valoración. Entonces:*

$$\hat{v}(\psi[\varphi/p]) = \widehat{v[\hat{v}(\varphi)/p]}(\psi)$$

Demostración. Procedemos por inducción estructural. Dividimos las proposiciones atómicas en cuatro casos como en la definición 1.35:

1. Si $\psi = p$ entonces, por definición de sustitución $\psi[\varphi/p] = \varphi$. Por tanto, $\hat{v}(\psi[\varphi/p]) = \hat{v}(\varphi)$ y siguiendo la notación que hemos explicado arriba, $\widehat{v[\hat{v}(\varphi)/p]}(\psi) = \widehat{v[\hat{v}(\varphi)/p]}(p) = v[\hat{v}(\varphi)/p](p) = \hat{v}(\varphi)$.

2. Si $\psi = q \in SP$ y $q \neq p$ entonces $\psi[\varphi/p] = q$, con lo que $\hat{v}(\psi[\varphi/p]) = \hat{v}(q)$ y, como antes, $\widehat{v[\hat{v}(\varphi)/p]}(q) = v[\hat{v}(\varphi)/p](q) = \hat{v}(q)$.

3. Si $\psi = \top$, $\psi[\varphi/p] = \top$, y $\widehat{v[\hat{v}(\varphi)/p]}(\top) = v(\top) = V$.

4. Si $\psi = \bot$, $\psi[\varphi/p] = \bot$, y $\widehat{v[\hat{v}(\varphi)/p]}(\bot) = v(\bot) = F$.

5. Si $\psi = \neg\psi_1$, entonces $\hat{v}((\neg\psi_1)[\varphi/p]) = \hat{v}(\neg(\psi_1[\varphi/p])) = v_\neg(\hat{v}(\psi_1[\varphi/p]))$, por las definiciones de sustitución y extensión de la valoración v. Aplicando la hipótesis de inducción, lo anterior es igual a

$$v_\neg(\widehat{v[\hat{v}(\varphi)/p]})(\psi_1) = \widehat{v[\hat{v}(\varphi)/p]}(\neg\psi_1).$$

6. Si $\psi = \psi_1 \square \psi_2$, entonces

$$\hat{v}(\psi[\varphi/p]) = \hat{v}((\psi_1[\varphi/p] \square \psi_2[\varphi/p])) = v_\square(\hat{v}(\psi_1[\varphi/p]),\ \hat{v}(\psi_2[\varphi/p])).$$

Aplicando la hipótesis de inducción, esto es igual a

$$v_\square(\widehat{v[\hat{v}(\varphi)/p]}(\psi_1),\ \widehat{v[\hat{v}(\varphi)/p]}(\psi_2)) = \widehat{v[\hat{v}(\varphi)/p]}(\psi_1 \square \psi_2).$$

□

Lema 1.39. *(De sustitución) Sean $\varphi, \varphi', \psi \in PROP_{SP}$, $p \in SP$. Si $\varphi \sim \varphi'$, $\psi[\varphi/p] \sim \psi[\varphi'/p]$.*

Demostración. Sea v valoración. Entonces, al ser $\hat{v}(\varphi) = \hat{v}(\varphi')$, se obtiene que

$$\hat{v}(\psi[\varphi/p]) = \widehat{v[\hat{v}(\varphi)/p]}(\psi) = \widehat{v[\hat{v}(\varphi')/p]}(\psi) = \hat{v}(\psi[\varphi'/p])$$

Como esto sucede para cualquier valoración v, $\psi[\varphi/p] \sim \psi[\varphi'/p]$. □

Acabamos de ver que sustituir fórmulas equivalentes en una misma fórmula nos lleva a fórmulas equivalentes. A continuación vemos que sustituir la misma fórmula en fórmulas equivalentes da como resultado fórmulas equivalentes.

Lema 1.40. *(De sustitución) Sean $\varphi, \psi_1, \psi_2 \in PROP_{SP}$, $p \in SP$. Si $\psi_1 \sim \psi_2$, entonces $\psi_1[\varphi/p] \sim \psi_2[\varphi/p]$.*

Demostración. Sea v cualquier valoración. Tenemos que:

$$\hat{v}(\psi_1[\varphi/p]) = \widehat{v[\hat{v}(\varphi)/p]}(\psi_1) = \widehat{v[\hat{v}(\varphi)/p]}(\psi_2) = \hat{v}(\psi_2[\varphi/p])$$

Lo que nos da el resultado. □

El lema 1.40 nos permite probar una versión más general de las leyes de equivalencia lógica, 1.34, en las que en vez de símbolos de proposición podemos sustituir cualquier fórmula. En el siguiente ejemplo probamos la primera:

Ejemplo 1.41. Consideremos las dos fórmulas $\psi_1 = p \vee q$ y $\psi_2 = q \vee p$. Sabemos que son equivalentes, luego si consideramos $\varphi, \chi \in PROP_{SP}$ tenemos que, por el lema 1.40, $\psi_1[\varphi/p]$ y $\psi_2[\varphi/p]$, es decir, $\varphi \vee q$ y $q \vee \varphi$, son equivalentes. Aplicando una vez más 1.40 para sustituir q por χ, obtenemos que $\varphi \vee \chi$ y $\chi \vee \varphi$ son equivalentes.

Ejemplo 1.42. Supongamos que queremos demostrar la equivalencia de estas dos fórmulas:

- $p_1 \vee p_2 \vee p_3 \to p_4 \wedge (p_5 \vee p_6)$
- $\neg((p_4 \vee p_5) \wedge (p_4 \vee p_6)) \to \neg(p_1 \vee p_2 \vee p_3)$

Desde luego podríamos hacerlo usando tablas de verdad, pero está claro que tardaríamos demasiado. Una forma más sencilla de hacerlo usando las leyes de equivalencia y los lemas de sustitución es comprobar las dos equivalencias lógicas:

1. $p_1 \vee p_2 \vee p_3 \to p_4 \wedge (p_5 \vee p_6) \sim p_1 \vee p_2 \vee p_3 \to (p_4 \vee p_5) \wedge (p_4 \vee p_6)$
2. $p_1 \vee p_2 \vee p_3 \to (p_4 \vee p_5) \wedge (p_4 \vee p_6) \sim \neg((p_4 \vee p_5) \wedge (p_4 \vee p_6)) \to \neg(p_1 \vee p_2 \vee p_3)$

En efecto:

- 1 se obtiene aplicando el lema 1.39 con $\psi = p_1 \vee p_2 \vee p_3 \to p$, $\varphi_1 = p_4 \wedge (p_5 \vee p_6)$,$\varphi_2 = (p_4 \vee p_5) \wedge (p_4 \vee p_6)$.
- 2 se obtiene en dos pasos:
  Aplicando el lema 1.40 con $\psi_1 = p \to q$, $\psi_2 = \neg q \to \neg p$ y $p = p_1 \vee p_2 \vee p_3$, obtenemos: $p_1 \vee p_2 \vee p_3 \to q \sim \neg q \to \neg(p_1 \vee p_2 \vee p_3)$.
  Y aplicando de nuevo el lema 1.40 con $\psi_1 = p_1 \vee p_2 \vee p_3 \to q$, $\psi_2 = \neg q \to \neg(p_1 \vee p_2 \vee p_3)$ y $q = (p_4 \vee p_5) \wedge (p_4 \vee p_6)$, obtenemos 2.

1.7. Completitud funcional

En 1.20 definimos recursivamente una serie de asignaciones de verdad que correspondían a los símbolos de las conectivas lógicas, $\neg, \wedge, \vee, \to, \leftrightarrow$. En general, a cualquier tabla de verdad dada por $v_\$: Bool^n \to Bool$ podemos asignarle un símbolo de conectiva n-aria[8], $\$(p_1, \ldots, p_n)$[9]. Según esta definición, podemos interpretar los símbolos $\top, \bot$ como conectivas 0-arias, dadas por las tablas de verdad:

$\top$	$\bot$
V	F

Analizando con mayor detenimiento las conectivas binarias, es fácil observar que existen 16 de ellas distintas, una asociada a cada tabla de verdad[10]. Por ejemplo, podemos definir mediante tablas de verdad las conectivas:

[8] Decimos que una conectiva es n-aria si se escribe con n símbolos de proposición, por ejemplo, las conectivas $\wedge, \vee, \to, \leftrightarrow$ son binarias (2-arias) y la conectiva $\neg$ es 1-aria.

[9] Aunque sintácticamente escribiremos las conectivas como $\$(p_1, \ldots, p_n)$, semánticamente el valor de verdad de $\$(p_1, \ldots, p_n)$ solo dependerá de los valores de verdad de $p_1, \ldots, p_n$, no de $p_1, \ldots, p_n$, al igual que con las conectivas conocidas anteriormente.

[10] Decimos que dos conectivas n-arias son distintas cuando tienen distinta tabla de verdad. Para cada n, hay 2^{2^n} conectivas n-arias distintas.

p	q	$p \veebar q$	$p \uparrow q$	$p \downarrow q$
V	V	F	F	F
V	F	V	V	F
F	V	V	V	F
F	F	F	V	V

$\veebar$ se conoce como *disyunción excluyente*, mientras que $\uparrow$ y $\downarrow$ se llaman *NAND* y *NOR*, respectivamente.

Sin embargo, en cierto sentido, estas nuevas conectivas son redundantes, ya que podemos expresarlas en función de nuestras cinco conectivas iniciales:

- $p \veebar q \sim \neg(p \leftrightarrow q)$
- $p \uparrow q \sim \neg(p \wedge q)$
- $p \downarrow q \sim \neg(p \vee q)$

Es natural, por tanto, preguntarse si cualquier conectiva n-aria será expresable en función de nuestras conectivas, como lo son $\veebar, \downarrow$ y $\uparrow$. Para formalizar esta cuestión, introducimos dos nociones de gran importancia:

Definición 1.43. Una conectiva n-aria \$ dada por $v_{\$} : Bool^n \to Bool$ se dice expresable en términos de un conjunto de conectivas C si existe una fórmula φ que tiene por conectivas a elementos de C y por símbolos de proposición a $p_1, ..., p_n$, tal que $\varphi \sim \$(p_1, ..., p_n)$.

Definición 1.44. (Completitud funcional) Un conjunto de conectivas C se dice funcionalmente completo si cualquier conectiva \$ dada por $v_{\$} : Bool^n \to Bool$ es expresable en términos de C.

Ejemplo 1.45. Si C es funcionalmente completo, cualquier proposición φ con símbolos $p_1, \ldots, p_n$ y cualesquiera conectivas $\$_1, \ldots, \$_m$ es equivalente a una expresión con símbolos $p_1, \ldots, p_n$ y conectivas en C.

En efecto, φ tiene una tabla de verdad en función de $p_1, \ldots, p_n$, de modo que se puede expresar como una conectiva n-aria $\$_\varphi$ con esa misma tabla de verdad, es decir, $\$_\varphi(p_1, \ldots, p_n) \sim \varphi$. Ahora, como C es funcionalmente completo, $\$_\varphi$ es expresable en términos de C, por tanto hay una expresión ψ de conectivas elementos de C y símbolos de proposición $p_1, \ldots, p_n$ tal que $\psi \sim \$_\varphi \sim \varphi$.

Ahora ya podemos dar respuesta a la pregunta que nos formulamos antes:

Proposición 1.46. $\{\neg, \wedge, \vee\}$ *es funcionalmente completo.*

Demostración. Sea \$ conectiva n-aria. Procedamos por inducción sobre n.

Si $n = 1$, es fácil comprobar que tenemos cuatro conectivas posibles, y que éstas corresponden a la identidad (es decir, $\$(p) \sim p$), $\neg$, $\top \sim p \vee \neg p$ y $\bot \sim p \wedge \neg p$,

con lo que se verifica el enunciado.

Supongamos que es válido para $n-1$ y veámoslo para n. Sea $\$(p_1, ..., p_n)$ una conectiva dada por la siguiente tabla de verdad:

p_n	p_1	...	p_{n-1}	$\$(p_1, ..., p_n)$
V	...	...	...	$X(1)$
...	...	...	...	...
V	...	...	...	$X(2^{n-1})$
F	...	...	...	$X(2^{n-1}+1)$
...	...	...	...	...
F	...	...	...	$X(2^n)$

donde $X(i)$ denota el valor de verdad correspondiente a la fila i-ésima. De esta forma obtenemos dos tablas de verdad, cada una con 2^{n-1} filas: aquella en la que p_n toma V como valor de verdad (las primeras 2^{n-1} filas de la tabla anterior) y aquella en la que toma el valor F (las filas restantes). Llamamos $\$_1(p_1, ..., p_{n-1})$ y $\$_2(p_1, ..., p_{n-1})$ a las conectivas inducidas por estas dos tablas de verdad.

Entonces tenemos:

$$\$(p_1, ..., p_n) \sim (p_n \wedge \$_1(p_1, ..., p_{n-1})) \vee (\neg p_n \wedge \$_2(p_1, ..., p_{n-1})).$$

Aplicando la hipótesis de inducción, existen fórmulas φ_1, φ_2 construidas a partir de $\{\neg, \wedge, \vee\}$ tales que $\varphi_1 \sim \$_1(p_1, ..., p_{n-1})$ y $\varphi_2 \sim \$_2(p_1, ..., p_{n-1})$, con lo que obtenemos que $\$(p_1, ..., p_n)$ es expresable en términos de $\{\neg, \wedge, \vee\}$. □

Definición 1.47. (Formas normales) Decimos que una fórmula está en *Forma Normal Conjuntiva* (FNC) si es de la forma:

$$\bigwedge_{i \leq n} \bigvee_{j \leq m_i} \varphi_{ij}$$

y se dice que está en *Forma Normal Disyuntiva* (FND) si es de la forma:

$$\bigvee_{i \leq n} \bigwedge_{j \leq m_i} \varphi_{ij}$$

donde cada φ_{ij} es una proposición atómica o la negación de una proposición atómica [11].

Corolario 1.48.

[11] Ya veremos en un ejercicio que, para cada fórmula φ, existen dos fórmulas φ^c, φ^d en FNC y en FND, respectivamente, tales que $\varphi \sim \varphi^c$ y $\varphi \sim \varphi^d$.

1. $\{\neg, \wedge\}$ *es funcionalmente completo.*
2. $\{\neg, \vee\}$ *es funcionalmente completo.*
3. $\{\rightarrow, \bot\}$ *es funcionalmente completo.*
4. $\{\uparrow\}$ *es funcionalmente completo.*
5. $\{\downarrow\}$ *es funcionalmente completo.*

Demostración.

1. Sabemos, por 1.46, que $\{\neg, \wedge, \vee\}$ es funcionalmente completo. Basta ver, entonces, que $\vee$ es expresable en términos de $\neg$ y $\wedge$. Esto es evidente por las leyes de De Morgan (1.34):
$$p \vee q \sim \neg(\neg p \wedge \neg q).$$
2. Análogo al anterior, usando la otra ley de De Morgan:
$$p \wedge q \sim \neg(\neg p \vee \neg q).$$
3. Basta ver que $\neg, \vee$ son expresables en términos de $\rightarrow, \bot$. Esto se sigue de estas afirmaciones que se pueden comprobar con tablas de verdad:
$$\neg p \sim p \rightarrow \bot.$$
$$p \vee q \sim (p \rightarrow \bot) \rightarrow q.$$
4. Expresamos $\neg, \wedge$ en términos de $\uparrow$:
$$\neg p \sim p \uparrow p.$$
$$p \wedge q \sim (p \uparrow q) \uparrow (p \uparrow q).$$
5. Expresamos $\neg, \vee$ en términos de $\downarrow$:
$$\neg p \sim p \downarrow p$$
$$p \vee q \sim (p \downarrow q) \downarrow (p \downarrow q).$$

□

1.8. Método de los *tableaux*

Dado un conjunto finito de fórmulas Ψ, nos interesa encontrar un método sistemático para determinar si $Insat(\Psi)$. Este procedimiento nos permitirá también deducir si, dados un conjunto de fórmulas $\Phi = \{\varphi_1, ..., \varphi_n\}$ y una fórmula φ, $\Phi \vDash \varphi$, pues por 1.30 esto es equivalente a $Insat(\Phi \cup \{\neg\varphi\})$ o, alternativamente, a que $\bigwedge_{i=1}^{n} \varphi_i \wedge \neg\varphi \sim \bot$. Es decir, aplicaremos el método a $\Psi := \Phi \cup \{\neg\varphi\}$.

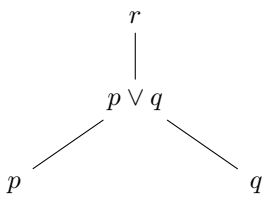
1.8.1. Nociones básicas y ejemplos

Nuestro objetivo es tomar los datos anteriores como vértices de un grafo y construir un *árbol*, es decir, un diagrama (llamado '*tableau*') en el que conectaremos las distintas fórmulas mediante aristas, formando *ramas* y de modo que, dados dos elementos del árbol, si pertenecen a la misma rama entonces se encuentran en conjunción y, si son de ramas distintas, se encuentran en disyunción. De esta forma, y a partir de una serie de reglas de construcción, el problema de la insatisfactibilidad de Ψ quedará reducido a comprobar que un *tableau* suyo tiene todas las ramas *cerradas*, esto es, contienen una contradicción.

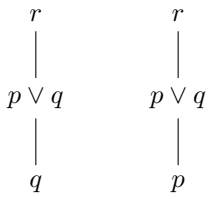
Construiremos nuestro diagrama de árbol, cuyos vértices serán proposiciones, de arriba hacia abajo. Llamamos al vértice de arriba del todo *raíz*. Llamaremos *hojas* a los vértices v tales que no haya ningún vértice v' debajo de v y unido por una arista a v. Llamaremos *rama* a una sucesión de vértices $\langle v_1, \ldots, v_n \rangle$ en que:

- v_1 es la raíz.
- v_{i+1} está unido por una arista a v_i y debajo de v_i para todo i.
- v_n es una hoja.

Por ejemplo, en el diagrama:



tendríamos estas dos ramas:



Al hacer el *tableau* podemos encontrarnos fórmulas que se pueden simplificar, llamadas *σ-fórmulas* o *simplificables*. Estas simplificaciones se podrán hacer de acuerdo a la siguiente tabla:

σ	σ_1
$\neg\top$	$\bot$
$\neg\bot$	$\top$
$\neg\neg\varphi$	φ

Está claro que, en cada fila, $\sigma \sim \sigma_1$.

Ahora bien, en el proceso de extender el *tableau*, nos conviene distinguir dos tipos de fórmulas, que llamaremos *α-fórmulas* y *β-fórmulas*. En las siguientes tablas, la primera columna servirá como definición de α-fórmulas y β-fórmulas, es decir, habrá cuatro tipos de α-fórmulas y cuatro tipos de β-fórmulas. En las otras dos columnas aparecerán sus descomposiciones, que usaremos al construir los *tableaux*.

- *α-fórmulas*:

α	α_1	α_2
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \to \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \to \varphi_2$	$\varphi_2 \to \varphi_1$

- *β-fórmulas*:

β	β_1	β_2
$\varphi_1 \vee \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\varphi_1 \to \varphi_2$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \to \varphi_2)$	$\neg(\varphi_2 \to \varphi_1)$

Podemos comprobar que para cada fila de la primera tabla, se cumple $\alpha \sim \alpha_1 \wedge \alpha_2$, y para cada fila de la segunda tabla, se cumple $\beta \sim \beta_1 \vee \beta_2$.

No es difícil probar el siguiente resultado, que justifica la anterior clasificación:

Proposición 1.49. *Sea $\varphi \in PROP_{SP}$. Entonces φ verifica una de las siguientes condiciones:*

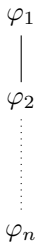
1. *Es de la forma p, $\neg p$, $\bot$ o $\top$, con $p \in SP$.*
2. *Es σ-fórmula.*
3. *Es α-fórmula.*
4. *Es β-fórmula.*

Debido a esto, nuestro método nos permitirá descomponer un conjunto de proposiciones múltiples veces hasta obtener proposiciones de tipo p_i, $\neg p_i$, $\bot$ o

$\top$, con $p_i \in SP$.

Ahora pasamos a describir las reglas de construcción de *tableaux* asociados a un conjunto Ψ de proposiciones.

- *Regla inicial*, R_{ini}: Sean $\varphi_1, ..., \varphi_n \in \Psi$. Entonces



  es *tableau* de Ψ.

- *Regla de hipótesis*, R_{hip}: Sea

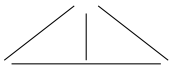
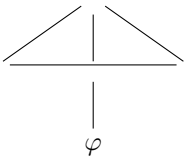


  tableau de Ψ. Entonces



  es *tableau* de Ψ, con $\varphi \in \Psi$.
  Es decir, siempre podemos añadir un elemento de Ψ a una rama.

- *Regla* σ, R_σ: Sea

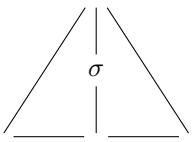
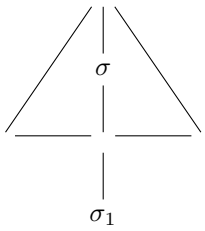


  tableau de Ψ, donde σ es fórmula simplificable y σ_1 la versión simplificada de σ. Entonces



es *tableau* de Ψ.
Es decir, siempre podemos añadir a una rama una versión simplificada de cualquier elemento de la rama y seguimos teniendo un *tableau* de Ψ.

- *Regla* α, R_α: Sea

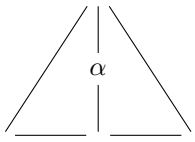
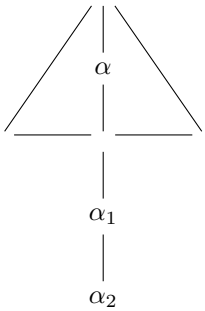


tableau de Ψ, donde α es una α-fórmula y α_1, α_2 su descomposición. Entonces



es *tableau* de Ψ.
Es decir, si hay una α-fórmula α en una rama, podemos añadir a la rama los dos elementos de su descomposición, α_1 y α_2.

- *Regla* β, R_β: Sea

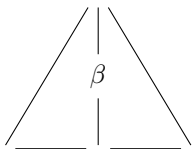
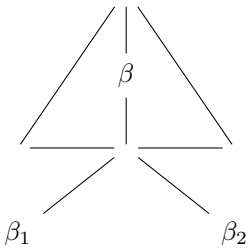


tableau de Ψ, donde β es una β-fórmula y β_1,β_2 su descomposición. Entonces



es *tableau* de Ψ.
Es decir, si hay una β-fórmula de descomposición β_1, β_2 en una rama, podemos dividir esa rama en dos ramificaciones, una que contiene a β_1 y otra que contiene a β_2.

Precisemos las nociones que describimos al inicio de esta sección. Dado un *tableau* T y una rama r, consideremos los siguientes conjuntos:

$$\Gamma_T := \{\varphi \in PROP_{SP} | \varphi \text{ está en } T\}$$

$$\Gamma_r := \{\varphi \in PROP_{SP} | \varphi \text{ está en } r\}$$

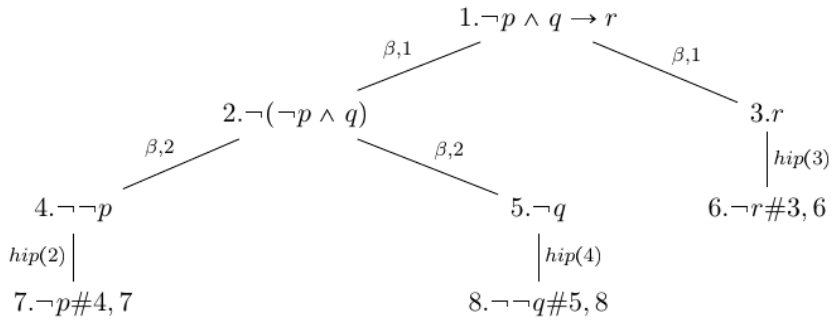
Esto nos permite establecer la siguiente

Definición 1.50. Una rama r de un *tableau* T se dice *cerrada* si se da al menos uno de los siguientes casos:

- $\bot \in \Gamma_r$.
- Existe φ tal que $\varphi, \neg\varphi \in \Gamma_r$.

Si todas las ramas de T son cerradas, T se dice *cerrado*. Si existe un *tableau* cerrado para $\Phi \cup \{\neg\varphi\}$ lo denotamos por $\Phi \vdash_{tb} \varphi$.
Decimos que una rama es *abierta* cuando no es cerrada, y decimos que un tablero es *abierto* cuando no es cerrada.

Ejemplo 1.51. Supongamos que queremos comprobar $\{\neg p \wedge q \to r, \neg p, \neg r\} \vDash \neg q$. Como veremos en breve, para ello nos basta encontrar un *tableau* cerrado para $\Psi = \{\neg p \wedge q \to r, \neg p, \neg r, \neg\neg q\}$. Enumeremos las premisas del siguiente modo: (1) $\neg p \wedge q \to r$, (2) $\neg p$, (3) $\neg r$, (4) $\neg\neg q$. La siguiente figura muestra un *tableau* que obtenemos de Ψ, y a continuación explicamos cómo se ha desarrollado a partir de las reglas de construcción:



Recordemos que el *tableau* se construye de arriba hacia abajo. Por la regla R_{ini}, el grafo de un solo vértice $\neg p \wedge q \to r$ es un *tableau* de Ψ, ya que $\neg p \wedge q \to r \in \Psi$.
Ahora bien, $\neg p \wedge q \to r$ es una β-fórmula de tipo $\beta = \varphi_1 \to \varphi_2$, de modo que, atendiendo a la tabla de las β-fórmulas, se descompone en las fórmulas $\beta_1 = \neg\varphi_1 = \neg(\neg p \wedge q)$ y $\beta_2 = \varphi_2 = r$.
Teniendo la descomposición de la β-fórmula, podemos aplicar la regla R_β, y obtenemos un *tableau* con dos ramificaciones, una con $\neg(\neg p \wedge q)$ y otra con r:

- Comenzamos con la segunda rama, que es la más sencilla. Usando la regla R_{hip}, podemos añadir a la rama cualquier elemento de Ψ, de modo que escogemos $\neg r$. Ahora bien, en esta rama tenemos las proposiciones $\neg p \wedge q \to r$, r y $\neg r$. En concreto, tenemos las proposiciones r y $\neg r$. Como en la rama hay dos proposiciones de tipo φ y $\neg\varphi$, es cerrada y hemos acabado con ella. Simbolizaremos que una rama está cerrada escribiendo # y señalando los vértices en los que se encuentran las proposiciones contradictorias.

- Fijémonos ahora la rama de $\neg(\neg p \wedge q)$. De nuevo, tenemos una β-fórmula. Usando la tabla como antes, vemos que se descompone en $\beta_1 = \neg\neg p$, $\beta_2 = \neg q$. De modo que, por la regla R_β, esta rama se divide a su vez en otras dos:

  - La primera rama tiene los elementos $\neg p \wedge q \to r$, $\neg(\neg p \wedge q)$ y $\neg\neg p$. Usando la regla R_{hip}, podemos añadir a esta rama el elemento $\neg p$ de Ψ. Ahora, dos de los elementos de esta rama son $\neg\neg p$ y $\neg p$. Es decir, tenemos las fórmulas φ y $\neg\varphi$, siendo $\varphi = \neg p$. Por tanto esta rama está cerrada.
  - La segunda rama tiene los elementos $\neg p \wedge q \to r$, $\neg(\neg p \wedge q)$ y $\neg q$. Usando la regla R_{hip}, podemos añadir a esta rama el elemento $\neg\neg q$ de Ψ. Ahora, dos de los elementos de esta rama son $\neg\neg q$ y $\neg q$. Como la rama contiene una fórmula y su negación, es cerrada.

Como todas las ramas son cerradas, el *tableau* es cerrado.

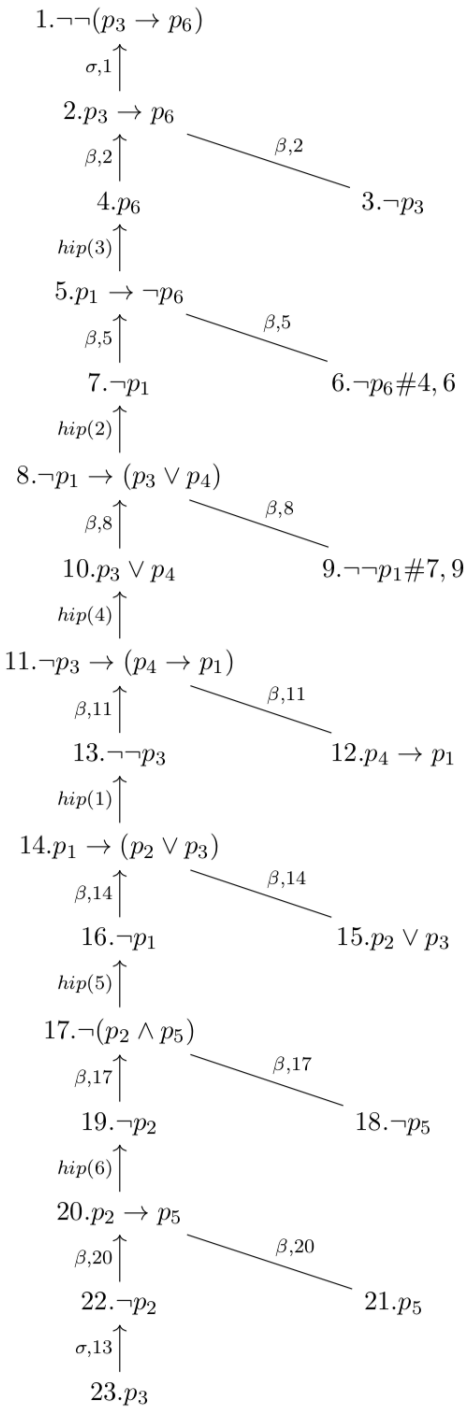
Vemos a continuación otro ejemplo en el que no se cumple que $\Phi \vDash \varphi$. Es decir, hay una valoración que cumple Φ y $\neg\varphi$. En estos casos, veremos que no se puede conseguir un *tableau* cerrado de $\Phi \cup \{\neg\varphi\}$. Al intentar obtener un *tableau* cerrado, la complicación del método aumentará considerablemente pero también nos dará indicaciones sobre qué valoración de verdad satisface a Φ y $\neg\varphi$.

Ejemplo 1.52. Sea $\varphi = \neg(p_3 \to p_6)$ y Φ formado por estas seis proposiciones:

1. $p_1 \to (p_2 \vee p_3)$
2. $\neg p_1 \to (p_3 \vee p_4)$
3. $p_1 \to \neg p_6$
4. $\neg p_3 \to (p_4 \to p_1)$
5. $\neg(p_2 \wedge p_5)$
6. $p_2 \to p_5$

En la siguiente figura intentamos conseguir un *tableau* cerrado de $\Phi \cup \{\neg\varphi\}$.

Como podemos ver, no conseguimos cerrar las ramas. En la rama que hemos indicado con flechas, no parece haber forma de encontrar una contradicción, y además tenemos en esta rama las proposiciones $\neg p_1$, $\neg p_2$, p_3 y p_6. En este caso podemos imaginar que la rama no se puede cerrar y hay una valoración que cumple los elementos de esa rama. En efecto, solo tenemos 4 opciones en función de los valores de verdad que asociemos a p_4 y p_5, y encontramos que la valoración dada por $v(p_1) = v(p_2) = v(p_5) = F$,$v(p_3) = v(p_4) = v(p_6) = V$ cumple Φ y $\neg\varphi$. Es decir, $\Phi \nvDash \varphi$. Como veremos en breve, esto significa que no podemos encontrar un *tableau* cerrado para $\Phi \cup \{\neg\varphi\}$.


1.¬¬(p3 → p6)
σ,1
2.p3 → p6
β,2
β,2
3.¬p3
4.p6
hip(3)
5.p1 → ¬p6
β,5
β,5
6.¬p6#4, 6
7.¬p1
hip(2)
8.¬p1 → (p3 ∨ p4)
β,8
β,8
9.¬¬p1#7, 9
10.p3 ∨ p4
hip(4)
11.¬p3 → (p4 → p1)
β,11
β,11
12.p4 → p1
13.¬¬p3
hip(1)
14.p1 → (p2 ∨ p3)
β,14
β,14
15.p2 ∨ p3
16.¬p1
hip(5)
17.¬(p2 ∧ p5)
β,17
β,17
18.¬p5
19.¬p2
hip(6)
20.p2 → p5
β,20
β,20
21.p5
22.¬p2
σ,13
23.p3

El último ejemplo muestra que hay muchas formas de intentar conseguir un *tableau* cerrado para un conjunto de fórmulas. Además, puede que dos formas distintas den lugar a *tableaux* con distinto número de proposiciones. Para obtener *tableaux* pequeños, nos conviene usar las reglas de construcción con la siguiente prioridad:

1. R_σ
2. R_α
3. R_β

Es decir, nos conviene aplicar siempre que sea posible la regla R_σ. Cuando no sea posible, aplicaremos la regla R_α. Si esta tampoco es posible, usaremos R_β.

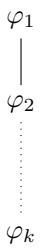
1.8.2. Corrección y completitud

Recordemos que lo que motivó esta sección fue, dados un conjunto de fórmulas Φ y una fórmula φ, el demostrar que $\Phi \vDash \varphi$, usando el método de los *tableaux*. Por tanto, para que el método nos permita alcanzar este objetivo tiene que cumplir dos propiedades:

- *Completitud:* $\Phi \vDash \varphi$ implica $\Phi \vdash_{tb} \varphi$. Es decir, si se cumple que $\Phi \vDash \varphi$, entonces podemos encontrar un *tableau* cerrado para $\Phi \cup \{\neg\varphi\}$.
- *Corrección:* $\Phi \vdash_{tb} \varphi$ implica $\Phi \vDash \varphi$. Es decir, si encontramos un *tableau* cerrado para $\Phi \cup \{\neg\varphi\}$, entonces se cumple que $\Phi \vDash \varphi$.

El siguiente resultado condensa las ideas expuestas hasta ahora:

Teorema 1.53. *Sean $\Phi \subseteq PROP_{SP}$ tal que $Sat(\Phi)$ y T tableau finito para Φ. Entonces existe una rama de T, r, tal que $Sat(\Gamma_r)$. Es más: si una valoración v cumple $v \vDash \Phi$, hay una rama r que cumple $v \vDash \Gamma_r$.*

Demostración. Todo *tableau* finito T se construye partiendo de 

y aplicando reglas de construcción de *tableaux* $R_1, ..., R_n$. Nos interesa hacer inducción sobre el número de reglas empleadas, n.

Así pues, sea una valoración v tal que $v \vDash \Phi$.

Si $n = 0$, por la regla inicial tenemos una única rama r tal que $\Gamma_r = \{\varphi_1, ..., \varphi_k\} \subseteq \Phi$. Como $v \vDash \Phi$, tenemos que $v \vDash \Gamma_r$.

Ahora sea $n > 0$, supongamos que el resultado se cumple para $n - 1$.

Sea Φ satisfactible, y sea T un *tableau* obtenido aplicando la regla inicial y otras reglas $R_1, \ldots, R_n$. Es decir, T se obtiene aplicando la regla R_n al *tableau* T' obtenido aplicando las reglas $R_1, \ldots, R_{n-1}$. Por hipótesis de inducción, tenemos una rama r de T' tal que $v \vDash \Gamma_r$. Al aplicar la regla R_n extendemos una rama de T' para obtener T.

Si la rama que extendemos no es r, entonces r también es una rama de T, por tanto, como dice el enunciado, $v \vDash \Gamma_r$ para una rama r de T.

Si la rama que extendemos es r, distinguimos los siguientes casos:

- $R_n = R_\sigma$. Para cierta fórmula $\sigma \in \Gamma_r$, añadimos σ_1 tal que $\sigma \sim \sigma_1$. Por tanto, como $v \vDash \Gamma_r$, en particular $v \vDash \sigma$ y así $v \vDash \sigma_1$. Entonces $v \vDash \Gamma_r \cup \{\sigma_1\}$, es decir, la rama r' formada al añadir σ_1 a r es la buscada.
- $R_n = R_\alpha$. Para cierta fórmula $\alpha \in \Gamma_r$, añadimos α_1, α_2 tales que $\alpha \sim \alpha_1 \wedge \alpha_2$. Por tanto, como $v \vDash \Gamma_r$, en particular $v \vDash \alpha$ y así $v \vDash \alpha_1$ y $v \vDash \alpha_2$. Entonces $v \vDash \Gamma_r \cup \{\alpha_1, \alpha_2\}$, es decir, la rama r' formada al añadir α_1 y α_2 a r es la buscada.
- $R_n = R_\beta$. Para cierta fórmula $\beta \in \Gamma_r$, añadimos β_1, β_2 tales que $\beta \sim \beta_1 \vee \beta_2$, formando dos nuevas ramas. Como $v \vDash \Gamma_r$, en particular $v \vDash \beta$. Esto quiere decir que o bien $v \vDash \beta_1$ o bien $v \vDash \beta_2$. En el primer caso, $v \vDash \Gamma_r \cup \{\beta_1\}$, por tanto la rama r_1 formada añadiendo β_1 a r cumple que $v \vDash \Gamma_{r_1}$. En el segundo caso, $v \vDash \Gamma_r \cup \{\beta_2\}$, por tanto la rama r_2 formada añadiendo β_2 a r cumple que $v \vDash \Gamma_{r_2}$. En ambos casos, tenemos una rama r' de T tal que $v \vDash r'$, por tanto se cumple el enunciado.
- $R_n = R_{hip}$. Añadimos a la rama r cierta $\varphi \in \Phi$. Por hipótesis, tenemos que $v \vDash \Phi$ y $v \vDash \Gamma_r$, luego en especial $v \vDash \Gamma_r \cup \{\varphi\}$. Es decir, la rama r' formada al añadir a r la hipótesis cumple $v \vDash r'$.

□

Corolario 1.54. *Sea $\Phi \subseteq PROP_{SP}$. Si Φ tiene un tableau cerrado, entonces $Insat(\Phi)$.*

Demostración. Se trata de la forma contrapositiva del enunciado 1.53. □

Corolario 1.55. *(Corrección) Sean $\Phi \in PROP_{SP}, \varphi \in PROP_{SP}$. Si $\Phi \vdash_{tb} \varphi$, entonces $\Phi \vDash \varphi$.*

Demostración. Si $\Phi \vdash_{tb} \varphi$ entonces $\Phi \cup \{\neg\varphi\}$ tiene un *tableau* cerrado. Por 1.54 se sigue que $Insat(\Phi \cup \{\neg\varphi\})$, es decir, aplicando 1.30, $\Phi \vDash \varphi$. □

Ahora pasamos a estudiar la completitud. Para ello es necesario introducir previamente varias nociones:

Definición 1.56. $\Phi \subseteq PROP_{SP}$ se dice *coherente* si no existe φ tal que $\varphi, \neg\varphi \in \Phi$ y $\bot \notin \Phi$.

Definición 1.57. $\Phi \subseteq PROP_{SP}$ se dice *saturado* si, usando la notación de las σ, α y β-fórmulas:

- $\sigma \in \Phi$ implica que $\sigma_1 \in \Phi$.
- $\alpha \in \Phi$ implica que $\alpha_1, \alpha_2 \in \Phi$.
- $\beta \in \Phi$ implica que $\beta_1 \in \Phi$ o $\beta_2 \in \Phi$.

Por ejemplo, si a un *tableau* T no le podemos aplicar más reglas de forma que se añadan nuevas proposiciones al *tableau*, entonces Γ_T es saturado.

Definición 1.58. $\Phi \subseteq PROP_{SP}$ se dice *de Hintikka* si es coherente y saturado.

En relación a los *tableaux*, nos gustaría poder tratar la 'complejidad' de las distintas fórmulas. Esto nos lleva a la siguiente:

Definición 1.59. Definimos la norma $||.|| : PROP_{SP} \to \mathbb{N}$ dada recursivamente por:

- $||\varphi|| = 0$, para toda $\varphi \in AT$.
- $||\neg\varphi|| = 1 + ||\varphi||$.
- $||\varphi_1 \wedge \varphi_2|| = 1 + ||\varphi_1|| + ||\varphi_2||$.
- $||\varphi_1 \vee \varphi_2|| = 1 + ||\varphi_1|| + ||\varphi_2||$.
- $||\varphi_1 \to \varphi_2|| = 2 + ||\varphi_1|| + ||\varphi_2||$.
- $||\varphi_1 \leftrightarrow \varphi_2|| = 5 + 2||\varphi_1|| + 2||\varphi_2||$.

La siguiente proposición es fácil de demostrar por casos y justifica nuestro tratamiento de las descomposiciones de las σ, α y β-fórmulas como expresiones reducidas de las fórmulas originales:

Proposición 1.60. *Sean σ, α y β σ, α y β-fórmulas, respectivamente, con $\sigma \sim \sigma_1$, $\alpha \sim \alpha_1 \wedge \alpha_2$, $\beta \sim \beta_1 \vee \beta_2$. Entonces:*

1. $||\sigma|| > ||\sigma_1||$.
2. $||\alpha|| > ||\alpha_1||$ *y* $||\alpha|| > ||\alpha_2||$.
3. $||\beta|| > ||\beta_1||$ *y* $||\beta|| > ||\beta_2||$.

Definición 1.61. (Valoración de Hintikka) Sea $H \subseteq PROP_{SP}$ de Hintikka. Definimos la valoración $v_H : SP \to \{V, F\}$ como:

- $v_H(p) = V$, si $p \in H$.
- $v_H(p) = F$, si $\neg p \in H$.

En caso de que $p, \neg p \notin H$, va a ser para nuestros propósitos indiferente el valor que toma v_H. Para que v_H esté bien definida, podemos suponer por ejemplo $v_H(p) = V$ si $p, \neg p \notin H$.

Proposición 1.62. *Si $H \subseteq PROP_{SP}$ es de Hintikka, $v_H \vDash H$. En particular, H es satisfactible.*

Demostración. Tenemos que demostrar que para toda $\varphi \in H$, $v_H \vDash \varphi$. Para ello procedemos por inducción sobre $||\varphi||$. Es necesario distinguir dos casos base primero.

Si $||\varphi|| = 0$ entonces $\varphi \in AT$ y, al ser H coherente, $\varphi \in SP$ o $\varphi = \top$. Si se da lo primero, entonces se da el resultado por la definición de v_H. Lo segundo es trivial, porque toda valoración satisface a $\top$.

Si $||\varphi|| = 1$ entonces solo puede ocurrir, por la coherencia de H, que $\varphi = \neg p$, para $p \in SP$; o que $\varphi = \top \vee \bot$. En el primer caso obtenemos que $v_H(p) = F$ y por tanto que $\hat{v}_H(\neg p) = V$. En el segundo caso se tiene claramente que $\hat{v}_H(\top \vee \bot) = V$.

Sea φ con $||\varphi|| > 1$, supongamos que el resultado se cumple para todo $n < ||\varphi||$. Por la proposición 1.49 tenemos las siguientes posibilidades:

- $\varphi = \sigma$ es σ-fórmula, con $\sigma \sim \sigma_1$. Al ser H saturado, $\sigma_1 \in H$. Sabemos de 1.60 que $||\sigma|| > ||\sigma_1||$, luego podemos aplicar la hipótesis de inducción y así $\hat{v}_H(\sigma_1) = V$, de lo que se sigue que $\hat{v}_H(\sigma_1) = \hat{v}_H(\sigma) = V$.

- $\varphi = \alpha$ es α-fórmula, con $\alpha \sim \alpha_1 \wedge \alpha_2$. Al ser H saturado, $\alpha_1, \alpha_2 \in H$ y, por 1.60, $||\alpha|| > ||\alpha_1||$ y $||\alpha|| > ||\alpha_2||$. Aplicando la hipótesis de inducción, $\hat{v}_H(\alpha_1) = \hat{v}_H(\alpha_2) = V$, luego $\hat{v}_H(\alpha) = \hat{v}_H(\alpha_1 \wedge \alpha_2) = V$.

- $\varphi = \beta$ es β-fórmula, con $\beta \sim \beta_1 \vee \beta_2$. Por 1.60, $||\beta|| > ||\beta_1||$ y $||\beta|| > ||\beta_2||$. Además, como H es saturado, o bien $\beta_1 \in H$ y, por hipótesis de inducción, $\hat{v}(\beta_1) = V$, o bien $\beta_2 \in H$ y, por hipótesis de inducción, $\hat{v}(\beta_2) = V$. En ambos casos, $\hat{v}(\beta) = \hat{v}(\beta_1 \vee \beta_2) = V$.

□

Corolario 1.63. *Sean $\Phi \subseteq PROP_{SP}$, T tableau abierto de Φ. Si:*

1. *Existe una rama abierta de T, r, tal que $\Phi \subseteq \Gamma_r$.*

2. *Γ_r es de Hintikka.*

entonces Φ es satisfactible.

Demostración. Como Γ_r es de Hintikka, por 1.62 es satisfactible, por tanto como $\Phi \subseteq \Gamma_r$, Φ es satisfactible.

□

Esto motiva la siguiente definición:

Definición 1.64. Sean $\Phi \subseteq PROP_{SP}$, T *tableau* de Φ. T se dice *completo* si toda rama abierta de T, r, verifica:

- $\Phi \subseteq \Gamma_r$.
- Γ_r es de Hintikka.

Proposición 1.65. *Sea $\Phi \subseteq PROP_{SP}$ finito*[12]*. Entonces existe un tableau completo para Φ.*

Demostración. En el siguiente capítulo probaremos una generalización de este teorema.

□

Corolario 1.66. *(Completitud) Sea $\Phi \subseteq PROP_{SP}$ finito. Si $\Phi \vDash \varphi$ entonces $\varphi \vdash_{tb} \varphi$.*

Demostración. $\Phi \vDash \varphi$ equivale por 1.30 a $Insat(\Phi \cup \{\neg\varphi\})$. Se sigue de 1.65 que podemos considerar un *tableau* T completo para $\Phi \cup \{\neg\varphi\}$. Veamos que T no puede tener ramas abiertas:

Si T tiene una rama abierta r, por definición de *tableau* completo, $\Phi \cup \{\neg\varphi\} \subseteq \Gamma_r$ y Γ_r es de Hintikka. Pero entonces, por el corolario 1.63, $\Phi \cup \{\neg\varphi\}$ sería satisfactible, lo cual contradice el enunciado.

Por tanto el *tableau* es cerrado , es decir, $\Phi \vdash_{tb} \varphi$. □

[12]Se puede prescindir de esta hipótesis, es decir, el enunciado sigue siendo cierto si Φ es infinito.

2 | Lógica de primer orden

2.1. Introducción

En el anterior capítulo, construímos con éxito un lenguaje formal que nos permitía traducir frases informales del español a expresiones formales, además de formalizar los conceptos de implicación y equivalencia lógica. Sin embargo, se puede reprochar que la lógica proposicional es demasiado *simple* en el sentido siguiente:

Consideremos el silogismo compuesto por dos premisas 'Todos los hombres son mortales', 'Sócrates es un hombre' y la conclusión 'Sócrates es mortal'. En este caso, podríamos pensar que la derivación se trata de una del tipo $p \wedge (p \to q) \to q$. Pero, por otro lado, parece evidente que depende de elementos más básicos que los símbolos de proposición. Sería, entonces, más conveniente una formalización del tipo: 'Para todo x, si x es hombre entonces es mortal', 'Sócrates es hombre' luego 'Sócrates es mortal'. Hemos empleado los términos 'hombre', 'mortal' y 'para todo' en un sentido puramente formal. Como ocurría con las proposiciones, existen múltiples frases y expresiones informales distintas que corresponden al silogismo que acabamos de exponer.

A continuación vamos a definir, igual que en lógica proposicional, el alfabeto que usaremos para construir fórmulas en los lenguajes de primer orden. Pero además, en este caso, habrá múltiples lenguajes de primer orden, y cada uno tendrá unos ciertos elementos que lo caractericen, que resumimos en el concepto de 'signatura':

Definición 2.1. Una *signatura* S es una tupla $\langle Ct_S, Fn_S, Pd_S \rangle$ donde:

- Ct_S es el conjunto de símbolos de constante.
- Fn_S es el conjunto de símbolos de función con determinada aridad[1].
- Pd_S es el conjunto de símbolos de predicado con determinada aridad.

[1] Por ahora, nos referimos por *aridad* de un símbolo de función o de predicado como el número de argumentos que admite. Más adelante especificaremos lo que significa esta idea.

Dado un símbolo Γ de función o predicado, denotamos por $\Gamma|_n$ que es n-ario[2].

Ejemplo 2.2. La signatura $Nat := \langle\{0, 1, 2, ...\}, \{+|_2, s|_1\}, \{< |_2\}\rangle$ para los números naturales. Es decir:[3]

- Los símbolos de constante son $0, 1, 2, \ldots$, que usaremos para representar los números naturales.
- Los símbolos de función serán el símbolo de función binaria $+$, que usaremos para denotar la función 'suma', y el símbolo de función 1-aria, s, que usaremos para denotar la función 'sucesor'.
- Tenemos un símbolo de predicado 2-ario, $<$, que se traducirá como 'menor o igual que'.

Definición 2.3. Dada la signatura S, definimos el alfabeto asociado como:

$$A_S := Ct_S \cup Fn_S \cup Pd_S \cup Var \cup \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \top, \bot\} \cup \{(,)\} \cup \{\exists, \forall\} \cup \{\doteq\},$$

donde:

- Ct_S, Fn_S, Pd_S vienen dados por 2.1.
- Var son los símbolos de variable:
- $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \top, \bot, (,)$ son los símbolos de conectiva y los paréntesis, al igual que en lógica proposicional.
- $\exists$ (para todo) y $\forall$ (existe) son los *cuantificadores lógicos*. Llamamos a $\forall$ *cuantificador universal* y a $\exists$ *cuantificador existencial*.
- $\doteq$, el símbolo de igualdad.

Como ocurría con las proposiciones, nos interesa distinguir las expresiones del alfabeto anterior que están bien formadas. Para ello, primero necesitaremos definir los *términos*, que podemos interpretar como las expresiones que usaremos para nombrar objetos, y después las *fórmulas*, expresiones que usaremos para denotar afirmaciones sobre los objetos.

Definición 2.4. Dada la signatura S, el *conjunto de términos* de S, $TERM_S$, es el menor subconjunto de A_S^* que verifica:[4]

1. $Ct_S \subseteq TERM_S$.
2. $Var \subseteq TERM_S$.

[2]Nótese que podríamos haber definido los símbolos de constante de nuestra signatura empleando símbolos de función 0-arios. Además, un predicado 0-ario se corresponde con un símbolo de proposición, lo que nos recuerda que al lógica de primer orden es más expresiva que la proposicional.

[3]Las interpretaciones que damos a continuación a los símbolos tendrán sentido al estudiar semántica más adelante en este capítulo.

[4]Recordemos que A_S^* es el cierre de Kleene de A_S, como definimos en 1.4.

3. Si $f|_n \in Fn_S$ y $t_1, .., t_n \in TERM_S$, entonces $f(t_1, ..., t_n) \in TERM_S$.

Ejemplo 2.5. Siguiendo con la signatura Nat, algunos ejemplos de elementos de $TERM_{NAT}$ serían $0, 1, 2, x, y, z, +(s(0), s(1))$ y $s(+(x, 3))$.

La siguiente proposición nos da una definición constructiva de $TERM_S$:

Proposición 2.6. *Sea $S = \langle Ct_S, Fn_S, Pd_S\rangle$ signatura. Definimos los conjuntos:*

$$T_S^0 := Ct_S \cup Var$$

$$T_S^{n+1} := T_S^n \cup \bigcup_{k\in\mathbb{N}} \{f(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in T_S^n,\ f|_k \in Fn_S\}$$

Entonces $\bigcup_{i\in\mathbb{N}} T_S^i = TERM_S$.

Demostración. La demostración es análoga a la de 1.8. □

Ahora podemos construir fórmulas a partir de estos elementos básicos que ya hemos definido. Comenzamos por

Definición 2.7. Dada la signatura S, el *conjunto de fórmulas atómicas* de S, $FORMAT_S$, es el menor subconjunto de A_S^* que verifica:

1. Si $t_1, t_2 \in TERM_S$, $t_1 \doteq t_2 \in FORMAT_S$.
2. Si $R|_n \in Pd_S$ y $t_1, \ldots, t_n \in TERM_S$, $R(t_1, \ldots, t_n) \in FORMAT_S$.
3. $\top, \bot \in FORMAT_S$.

Definición 2.8. Dada la signatura S, el *conjunto de fórmulas* de S, $FORM_S$, es el menor subconjunto de A_S^* que verifica:

1. $FORMAT_S \subseteq FORM_S$.
2. Si $\varphi_1, \varphi_2 \in FORM_S$, $(\neg\varphi_1), (\varphi_1 \square \varphi_2) \in FORM_S$.
3. Si $x \in Var$ y $\varphi \in FORM_S$, $(Qx\,\varphi) \in FORM_S$, siendo $Q \in \{\forall, \exists\}$

A partir de ahora, usaremos para mayor brevedad el símbolo Q como intercambiable por $\forall$ o $\exists$ a no ser que se indique lo contrario.

Damos una definición constructiva de $FORM_S$:

Proposición 2.9. *Sea $S = \langle Ct_S, Fn_S, Pd_S\rangle$ signatura. Definimos los conjuntos:*

$$F_S^0 := FORMAT_S$$

$$F_S^{n+1} := F_S^n \cup \{(\neg\varphi) \mid \varphi \in F_S^n\} \cup \{(\varphi \square \psi) \mid \varphi, \psi \in F_S^n\} \cup \{(Qx\,\varphi) \mid x \in Var,\ \varphi \in F_S^n\}$$

Entonces $\bigcup_{i\in\mathbb{N}} F_S^i = FORM_S$.

Demostración. De nuevo, la demostración es análoga a la de 1.8. □

2.2. Inducción estructural y recursión

Tal y como ocurría con la lógica proposicional, la recursión y la inducción estructural son las dos principales herramientas empleadas en las demostraciones de la lógica de primer orden. Sin embargo, ahora tenemos que tratar separadamente los conjuntos $TERM_S$ y $FORM_S$ asociados a cierta signatura S.

Comenzamos con los teoremas de inducción estructural y recursión para $TERM_S$, dada una signatura $S = \langle Ct_S, Fn_S, Pd_S \rangle$:

Proposición 2.10. *(Inducción estructural) Sean $S = \langle Ct_S, Fn_S, Pd_S \rangle$ una signatura y P una propiedad. Si se cumple que:*

1. *P es válida para todo elemento de T_S^0.*
2. *Sea $f|_k \in Fn_S$. Si P es válida para todo elemento de T_S^n, entonces es válida para $f(t_1, \dots t_n)$, con $t_1, \dots, t_n \in T_S^n$.*

Entonces P es válida para todo elemento de $TERM_S$.

Demostración. Si la propiedad P cumple 1, 2 y 3, entonces el conjunto de términos que cumplen P cumple las tres propiedades de la definición 2.4, por tanto contiene a $TERM_S$. □

Proposición 2.11. *El esquema de definición recursiva da como resultado una única función, es decir, dadas:*

1. *$F_0 : Ct_S \cup Var \to A$.*
2. *$F_f : A^k \to A$, para cada función $f|_k \in Fn_S$.*

existe una única función $F : TERM_S \to A$ tal que:

1. *$F(t) = F_0(t)$, para todo $t \in Ct_S \cup Var$.*
2. *$F(f(t_1, \dots, t_k)) = F_f(F(t_1), \dots, F(t_k))$, para cada $f|_k \in Fn_S$ y $t_1, \dots, t_k \in TERM_S$.*

Demostración. Se omite la demostración. □

Ejemplo 2.12. Una función que nos será de utilidad será var, que nos lleva cada elemento de $TERM_S$ al conjunto de variables que aparecen en él. La definimos recursivamente como:

1. Caso base: $var_0 : Ct_S \cup Var \to \mathcal{P}(Var)$, dada por $var_0(c) = \emptyset$ si $c \in Ct_S$ y $var(x) = \{x\}$ si $x \in Var$.
2. Caso recursivo: Dado $f|_k \in Fn_S$, $var_f : (\mathcal{P}(Var))^k \to \mathcal{P}(Var)$, dada por $var_f(f(t_1, \dots t_k)) = \bigcup_{i=1}^{k} var(t_i)$.

Ahora enunciamos los teoremas de inducción estructural y recursión para $FORM_S$, dada una signatura $S = \langle Ct_S, Fn_S, Pd_S \rangle$:

Proposición 2.13. *(Inducción estructural) Sean $S = \langle Ct_S, Fn_S, Pd_S \rangle$ una signatura y P una propiedad. Si se cumple que:*

1. *P es válida para todo elemento de $FORMAT_S$.*
2. *Si $\varphi_1, \varphi_2 \in FORM_S$ y se cumplen $P(\varphi_1), P(\varphi_2)$, entonces tenemos $P((\neg\varphi_1))$ y $P((\varphi_1 \square \varphi_2))$.*
3. *Si $x \in Var$ y $\varphi \in FORM_S$, y se cumple $P(\varphi)$, entonces tenemos $P((Qx\varphi))$.*

Entonces P es válida para todo elemento de $FORM_S$.

Demostración. Si la propiedad P cumple 1, 2 y 3, entonces el conjunto de términos que cumplen P cumple las tres propiedades de la definición 2.8, por tanto contiene a $FORM_S$. □

Proposición 2.14. *El esquema de definición recursiva da como resultado una única función, es decir, dadas:*

1. *$F_{AT} : FORMAT_S \to A$.*
2. *$F_\neg : A \to A$.*
3. *$F_\square : A \times A \to A$.*
4. *$F_Q : Var \times A \to A$.*

existe una única función $F : FORM_S \to A$ tal que:

1. *$F(\varphi) = F_{AT}(\varphi)$, para toda $\varphi \in FORMAT_S$.*
2. *$F((\neg\varphi)) = F_\neg(F(\varphi))$.*
3. *$F((\varphi_1 \square \varphi_2)) = F_\square(F(\varphi_1), F(\varphi_2))$.*
4. *$F((Qx\,\varphi)) = F_Q(x, F(\varphi))$.*

Demostración. Se omite la demostración. □

Ejemplo 2.15. Extendamos la función *var* al conjunto $FORM_S$. Por comodidad, omitimos las funciones auxiliares:

1. Caso base:
   - $var(\top) = var(\bot) = \emptyset$.
   - Sean $p|_k \in Pd_S$, $t_1, \ldots t_n \in TERM_S$. Entonces $var(p(t_1, \ldots t_n)) = \bigcup_{i=1}^{k} var(t_i)$.
   - Sean $t, s \in TERM_S$. Entonces $var(t \doteq s) = var(t) \cup var(s)$.

2. Caso recursivo:

   - $var(\neg\varphi) = var(\varphi)$.
   - $var(\phi \square \psi) = var(\varphi) \cup var(\psi)$.
   - $var(Qx\,\varphi) = \{x\} \cup var(\varphi)$.

2.3. Variables libres y ligadas

Consideremos las siguientes fórmulas de primer orden:

- $(\forall x\ x \doteq 3)$
- $x \doteq 3$

La primera fórmula se puede traducir informalmente como 'para todo x, x es igual a 3'. Intuitivamente, podemos decir que el cuantificador $\forall$ está afectando al significado de x. Sin embargo, en la segunda fórmula, x no aparece afectada por ningún cuantificador. En general, diremos que una variable es *ligada* en una fórmula φ si siempre aparece afectada por un cuantificador, como en el primer ejemplo, y diremos que es una variable *libre* en φ si aparece alguna vez sin estar afectada por un cuantificador. Formalicemos estos conceptos:

Definición 2.16. Sea $S = \langle Ct_S, Fn_S, Pd_S \rangle$ signatura. Definimos recursivamente la función $lib : FORM_S \to \mathcal{P}(Var)$, que nos lleva cada fórmula al conjunto de sus variables libres:

1. Caso base: Sea $\varphi \in FORMAT_S$. Entonces, $lib(\varphi) = var(\varphi)$.
2. Caso recursivo:

   - $lib(\neg\varphi) = lib(\varphi)$.
   - $lib(\varphi \square \psi) = lib(\varphi) \cup lib(\psi)$.
   - $lib(Qx\,\varphi) = lib(\varphi) \setminus \{x\}$.

Definición 2.17. Sea $S = \langle Ct_S, Fn_S, Pd_S \rangle$ signatura. Definimos el *conjunto de sentencias*, $SENT_S$, como el formado por aquellas $\varphi \in FORM_S$ tales que $lib(\varphi) = \emptyset$.