Fundamentos Lógicos de la Informática

Bibliografía:

- Diapositivas de Fundamentos Lógicos de la Informática, Departamento de Ingeniería de al Información y las Comunicaciones (nota: la errata es de ellos), Facultad de Informática, Universidad de Murcia.
- Wikipedia, la enciclopedia libre (https://es.wikipedia.org/).

Capítulo 1

Lógica

La lógica estudia las oraciones y los razonamientos, y existen tantas como tipos de oraciones y razonamientos. En informática, es la base de la programación, representa el conocimiento en inteligencia artificial, sirve para demostraciones de resultados teóricos y diseño de circuitos lógicos y define los problemas NP-completos (SAT). Existen distintas lógicas, como las lógicas clásicas (proposicional, categórica, de primer orden...) y la lógica difusa.

1.1. Oraciones

Dentro de una misma lógica, una oración lógica es una oración del lenguaje natural que cumpla ciertas condiciones. En las lógicas clásicas, es aquella que es **enunciativa** y cumple la **ley del tercero excluido** (solo puede ser verdadera [V] o falsa [F]) y la **ley de no contradicción** (no puede ser V y F a la vez). Estas pueden ser **simples** (**atómicas**) o **compuestas** (**moleculares**), dependiendo de si tienen uno o más predicados.

1.2. Razonamientos

Un razonamiento es una estructura que enlaza oraciones, de las cuales una es la **conclusión** de otras (**premisas**) y todas (salvo ella misma) proporcionan evidencias para justificarla. Así, un razonamiento está formado por **axiomas** o **premisas**; **conclusiones** o **teoremas**, y una **demostración**. Existen dos tipos de razonamiento:

- **Deductivo:** Se basa en la implicación.
- **Inductivo:** Parte de casos y llega a una conclusión general. Sólo es válido si se consideran todos los casos; de lo contrario la conclusión es probablemente, pero no necesariamente, cierta.

Un razonamiento es válido si la conclusión es necesariamente cierta cuando lo son las premisas, y se escribe como $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vDash \underset{\text{Conclusión}}{\beta}$. Para representarlo:

- **Representación gráfica:**

$$\begin{array}{l} P_1 \quad \wedge \quad P_2 \\ \hline \quad \downarrow \\ \quad C_1 \qquad \wedge \quad P_3 \\ \hline \qquad\qquad \downarrow \\ \qquad\qquad C_2 \end{array}$$

- **Representación estándar:**

$$\begin{array}{lll} & P_1 & \\ & P_2 & \\ \hline \therefore & C_1 & P_1 \textbf{ y } P_2 \\ & P_3 & \\ \hline \therefore & C_2 & C_1 \textbf{ y } P_3 \end{array}$$

1.3. Tipos de definiciones

- **Extensiva** o **extensional:** Lista de elementos que cumplen la condición.
- **Comprensiva** o **intensional:** Lista de propiedades necesarias.
- **Recursiva:** Formada por una **regla base**, que define casos concretos, y una **regla recursiva**, que define todos los demás casos a partir de casos ya conocidos mediante una relación. También puede contener una **regla de exclusión**.

1.4. Lenguajes formales

El lenguaje natural es ambiguo, y suele ser vago, paradójico, complicado... Por tanto, en ciencia es imprescindible un lenguaje formal para obtener rigor. Un lenguaje formal consta de un conjunto de símbolos (**alfabeto** o **vocabulario**) y una definición recursiva para conectarlos (**gramática** o **sintaxis**), y es el conjunto de todas las fórmulas bien formadas (f.b.f.) obtenidas a partir de estas. En la práctica, necesitamos un sistema de codificación (formalización) y de interpretación.

1.5. Formalización e interpretación

Formalizar es obtener una oración o f.b.f. en lenguaje formal a partir del lenguaje natural, mientras que interpretar es entender una f.b.f. expresándola en lenguaje natural.

Capítulo 2

Lógica proposicional

Las oraciones lógicas en lógica proposicional (**L0**) se llaman **proposiciones**. Las proposiciones atómicas, también llamadas **sentencias** o **átomos**, se agrupan mediante **operadores lógicos** para formar oraciones compuestas.

2.1. Sintaxis

- **Constantes:** Verdadero (V) o falso (F). $\mathbb{B} = \{V, F\}$.
- **Sentencias:** Se representan por un conjunto de letras latinas. El conjunto de todos se denota por $\mathcal{P}$.
- **Operadores lógicos:** Negación ($\neg$) y conectivos. Los conectivos son: conjunción ($\wedge$), disyunción ($\vee$), implicación ($\rightarrow$) y doble implicación ($\leftrightarrow$).
- **Paréntesis** o corchetes, para agrupar expresiones.

Definición recursiva de una f.b.f.:

- **Forma básica:** Todo átomo es una f.b.f.
- **Forma recursiva:** Si α y β son f.b.f., también lo son $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ y $(\alpha \leftrightarrow \beta)$. La presencia o ausencia de paréntesis es importante.

En la práctica, podemos eliminar paréntesis según estas reglas:

- Se pueden eliminar los paréntesis exteriores.
- **Prioridad:** De mayor a menor: $\neg$, ($\wedge$, $\vee$), ($\rightarrow$, $\leftrightarrow$).
- **Asociatividad:** A igual prioridad de operadores, se asocia por la izquierda.

También podemos añadir paréntesis a cualquier expresión que no sea una negación.

2.2. Formalización

- Los átomos corresponden a oraciones enunciativas afirmativas, en forma presente y con sujeto (salvo verbos impersonales).
- $\neg\alpha$: No α, no es el caso de α, no es cierto que α, es falso que α, no sucede que α, la negación de α.
- $\alpha \wedge \beta$: α y β (pero, aunque, además, sin embargo, también, a la vez, aún, no obstante).
- $\alpha \vee \beta$: O α o β; ya α, ya β, ya ambas.
- $\alpha \to \beta$: Si α β, si α entonces β, α solo si β, solo α si β, es suficiente α para que β, siempre que α entonces β, no α a menos que β, es necesario β para que α, a no ser que β no α.
- $\alpha \leftrightarrow \beta$: α si y sólo si β, α equivale a β, α cuando y sólo cuando β, α cuando únicamente β , α es condición suficiente y necesaria para que β.

2.3. Interpretación

Procedimiento que traduce las fórmulas **atómicas** a oraciones naturales. Una **asignación** v_I es el procedimiento que establece un valor de verdad a una fórmula atómica según una interpretación I. En L0 no se suele hacer distinción, y hace referencia a una función $v_I : \mathcal{P}_\alpha \to \mathbb{B}$ tal que $V \mapsto V$ y $F \mapsto F$.

La **evaluación** es la obtención del valor de verdad de una oración α. Decimos $V(\alpha) = V$ o $V(\alpha) = F$, según corresponda.

- **Regla base:** Si $\alpha \in \mathcal{P}$, entonces $V(\alpha) = v_I(\alpha)$.
- **Regla recursiva:**

$$
\begin{aligned}
V(\neg\alpha) &= \begin{cases} V & \text{si } V(\alpha) = F \\ F & \text{si } V(\alpha) = V \end{cases} \\
V(\alpha \wedge \beta) &= \begin{cases} V & \text{si } V(\alpha) = V \text{ y } V(\beta) = V \\ F & \text{en otro caso} \end{cases} \\
V(\alpha \vee \beta) &= \begin{cases} F & \text{si } V(\alpha) = F \text{ y } V(\beta) = F \\ V & \text{en otro caso} \end{cases} \\
V(\alpha \to \beta) &= \begin{cases} F & \text{si } V(\alpha) = V \text{ y } V(\beta) = F \\ V & \text{en otro caso} \end{cases} \\
V(\alpha \leftrightarrow \beta) &= \begin{cases} V & \text{si } V(\alpha) = V(\beta) \\ F & \text{en otro caso} \end{cases}
\end{aligned}
$$

2.4. Grafos semánticos

Un grafo semántico es un árbol que representa una f.b.f. El nodo principal contiene el operador principal (o el único átomo). De cada conectivo parten dos ramas (o una si es $\neg$) con las subfórmulas que conecta, y los átomos son hojas.

2.5. Decidibilidad

Una oración puede ser:

- **Satisfacible** si $V(\alpha) = V$ en alguna interpretación.
- **Falseable** si $V(\alpha) = F$ en alguna interpretación.
- **Contingente** o **contingencia** si es a la vez satisfacible y falseable.
- **Tautológica**, **válida** o **tautología** si $V(\alpha) = V$ en todas las interpretaciones. Escribimos $\models \alpha$.
- **Insatisfacible** o **contradicción** si $V(\alpha) = F$ en todas las interpretaciones.

El problema SAT, determinar si una oración lógica es satisfacible, es el primer problema conocido NP-completo, y de hecho todos los problemas NP-completos se pueden reducir a SAT, de modo que si uno de resuelve como P, se resuelven todos.

Un conjunto de fórmulas $\mathcal{F} = \{\alpha_1, \ldots, \alpha_n\}$ es satisfacible si su conjunción lo es, y llamamos **modelo** de $\mathcal{F}$ a cualquier interpretación en la que $V(\alpha_1 \wedge \cdots \wedge \alpha_n) = V$. Definimos del mismo modo conjunto insatisfacible. El conjunto $\mathcal{F} = \{\}$ es modelo en todas las interpretaciones.

Para hallar los valores de verdad de una oración en función de la interpretación, podemos construir una **tabla de verdad.** Si α tiene n átomos y m operadores, construimos una tabla con 2^n filas (más la cabecera) y $n + m$ columnas. En cada fila establecemos una asignación hasta establecer todas las asignaciones posibles y obtenemos las evaluaciones para las oraciones definidas por cada operador, en orden de evaluación y terminando con el operador principal, que establece el valor de verdad. Debemos indicar el orden de evaluación.

Otra forma es la **propagación de literales.** Un literal es un átomo o la negación de un átomo. Dada una fórmula ϕ, definimos $\phi(p) \equiv \phi_{|V(p)=V}$ a la fórmula más simplificada que, en las interpretaciones en las que $V(p) = V$, tenga los mismos valores de verdad que ϕ. Por ejemplo, dada la oración $\phi \equiv (p \to q) \to (\neg p \to \neg q)$, tendríamos que $\phi(p) \equiv \phi_{|V(p)=V} \equiv (V \to q) \to (\neg V \to \neg q) \equiv q \to (F \to q) \equiv q \to V \equiv V$, mientras que $\phi(\neg p) \equiv \phi_{|V(\neg p)=V} \equiv \phi_{|V(p)=F} \equiv (F \to q) \to (\neg F \to \neg q) \equiv V \to (V \to \neg q) \equiv V \to \neg q \equiv \neg q$. En el segundo caso, tendríamos, por ejemplo, que $\phi(\neg p)(q) \equiv \phi(\neg p, q) \equiv \phi(\neg p)_{|V(q)=V} \equiv \neg V \equiv F$. En la práctica bastaría con escribir $\phi(\neg p, q) \equiv \neg V \equiv F$ para este último caso.

Para comprobar los valores de verdad realizaríamos un **árbol semántico.** En este, la raíz sería la fórmula inicial, y de cada nodo, que contendrá una fórmula ξ, partirán dos ramas con $\xi(p)$ y $\xi(\neg p)$ para algún átomo l en ξ (normalmente el que más aparece), salvo si $\xi \equiv V$ o $\xi \equiv F$. A la hora de dibujarlo, la línea que une una expresión con otra derivada se etiqueta con el literal a propagar.

2.6. Equivalencias

Dos expresiones α y β son lógicamente equivalentes si y sólo si $V(\alpha) = V(\beta)$ para cualquier interpretación. Escribimos $\alpha \equiv \beta$. Así, $\alpha \equiv \beta \iff \vDash \alpha \leftrightarrow \beta$.

- **Propiedades conmutativas:** $\alpha \wedge \beta \equiv \beta \wedge \alpha$; $\alpha \vee \beta \equiv \beta \wedge \alpha$; $\alpha \leftrightarrow \beta \equiv \beta \leftrightarrow \alpha$.
- **Propiedades asociativas:** $\alpha \wedge (\beta \wedge \gamma) \equiv (\alpha \wedge \beta) \wedge \gamma$; $\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$; $\alpha \leftrightarrow (\beta \leftrightarrow \gamma) \equiv (\alpha \leftrightarrow \beta) \leftrightarrow \gamma$.
- **Propiedades de De Morgan:** $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$; $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$.
- **Propiedades distributivas:** $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$; $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$; $\alpha \rightarrow (\beta \vee \gamma) \equiv (\alpha \rightarrow \beta) \vee (\alpha \rightarrow \gamma)$; $\alpha \rightarrow (\beta \wedge \gamma) \equiv (\alpha \rightarrow \beta) \wedge (\alpha \rightarrow \gamma)$.
- **Propiedades de absorción:** $\alpha \vee (\alpha \wedge \beta) \equiv \alpha \wedge (\alpha \vee \beta) \equiv \alpha$.
- **Expresión booleana:** $\alpha \vee (\neg\beta \wedge \beta) \equiv \alpha \wedge (\neg\beta \vee \beta) \equiv \alpha$.
- **Reducción al absurdo:** $\neg\alpha \rightarrow (\beta \wedge \neg\beta) \equiv \alpha$.
- **Propiedad de contraposición** o **transposición:** $\alpha \rightarrow \beta \equiv \neg\beta \rightarrow \neg\alpha$.
- **Exportación:** $(\alpha \wedge \beta) \rightarrow \gamma \equiv \alpha \rightarrow (\beta \rightarrow \gamma)$.
- **Idempotencia:** $\alpha \equiv \neg(\neg\alpha) \equiv \alpha \vee \alpha \equiv \alpha \wedge \alpha$.
- **Eliminación del condicional:** $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta \equiv \neg(\alpha \wedge \neg\beta)$.
- **Eliminación del bicondicional:** $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha) \equiv (\alpha \wedge \beta) \vee (\neg\beta \wedge \neg\alpha)$.
- **Propiedades sobre tautologías:** $\alpha \vee \neg\alpha \equiv V$; $V \vee \beta \equiv V$; $V \wedge \beta \equiv \beta$.
- **Propiedades sobre insatisfacibilidad:** $\alpha \wedge \neg\alpha \equiv F$; $F \vee \beta \equiv \beta$; $F \wedge \beta \equiv F$.

2.7. Razonamientos válidos

Un razonamiento es válido si y sólo si en todas las interpretaciones en las que α es verdad, β también lo es. Igualmente, β es **consecuencia lógica** de $\mathcal{F} = \{\alpha_1, \ldots, \alpha_n\}$ si β es verdad siempre que $\mathcal{F}$ sea un modelo. Escribimos $\alpha \vDash \beta$ o $\mathcal{F} \vDash \beta$, y sabemos que $\alpha \vDash \beta \iff \vDash \alpha \rightarrow \beta$.

Teorema de la deducción semántica: $\mathcal{F} \cup \{\alpha\} \vDash \beta \iff \mathcal{F} \vDash \alpha \rightarrow \beta$. Corolario: $\mathcal{F} \vDash \beta \iff \vDash \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta \iff \vDash \neg(\alpha_1 \wedge \cdots \wedge \alpha_n \wedge \neg\beta) \iff \alpha_1 \wedge \cdots \wedge \alpha_n \wedge \neg\beta$ es insatisfacible. Propiedades generales de $\vDash$:

- **Reflexividad:** $\alpha \vDash \alpha$.
- **Transitividad:** Si $\mathcal{F} \vDash \alpha$ y $\alpha \vDash \beta$ entonces $\mathcal{F} \vDash \beta$.
- **Monotonía:** Si $\mathcal{F} \vDash \alpha$ entonces $\mathcal{F} \cup \{\beta\} \vDash \alpha$.
- Si $\mathcal{F} \vDash \alpha$ y $\vDash \beta$, entonces $\mathcal{F} \backslash \{\beta\} \vDash \alpha$.
- $\alpha \equiv \beta \iff \alpha \vDash \beta$ y $\beta \vDash \alpha$.

Algunas propiedades:

- **Simplificación** o **eliminación de la conjunción:** $\alpha \wedge \beta \vDash \alpha$.
- **Adición** o **introducción de la disyunción:** $\alpha \vDash \alpha \vee \beta$.
- **Silogismos:** Forma de razonamiento deductivo con dos premisas y una conclusión.
  - **Categóricos**
    - **Combinación** o **introducción de la conjunción:** $\{\alpha, \beta\} \vDash \alpha \wedge \beta$.
    - **Inconsistencia:** $\{\alpha, \neg\alpha\} \vDash \beta$.
  - **Hipotéticos**
    - **Silogismo hipotético:** $\{\alpha \to \beta, \beta \to \gamma\} \vDash \alpha \to \gamma$.
    - **Demostración por casos:** $\{\alpha \to \gamma, \beta \to \gamma\} \vDash \alpha \vee \beta \to \gamma$.
    - **Prueba por casos:** $\{\alpha \to \beta, \neg\alpha \to \beta\} \vDash \beta$.
  - **Hipotéticos mixtos**
    - **Modus Ponens:** $\{\alpha \to \beta, \alpha\} \vDash \beta$.
    - **Modus Tollens:** $\{\alpha \to \beta, \neg\beta\} \vDash \neg\alpha$.
  - **Disyuntivo:** $\{\alpha \vee \beta, \neg\beta\} \vDash \alpha$.
- **Dilemas:** Forma de razonamiento con una premisa disyunción que representa las opciones, normalmente contrarias.
  - **Constructivo:** $\{\alpha \vee \beta, \alpha \to \gamma, \beta \to \delta\} \vDash \gamma \vee \delta$.
  - **Destructivo:** $\{\neg\gamma \vee \neg\delta, \alpha \to \gamma, \beta \to \delta\} \vDash \neg\alpha \vee \neg\beta$.
  - **Transposición:** $\alpha \to \beta \vDash \neg\beta \to \neg\alpha$.
  - **Eliminación de la equivalencia:** $\alpha \leftrightarrow \beta \vDash \{\alpha \to \beta, \beta \to \alpha\}$.
  - **Introducción de la equivalencia:** $\{\alpha \to \beta, \beta \to \alpha\} \vDash \alpha \leftrightarrow \beta$.

El Corolario del Teorema de la Deducción Semántica y las propiedades básicas de equivalencia y razonamientos nos permiten considerar al menos dos estrategias de razonamiento deductivo: la **demostración directa**, comprobando que β es consecuencia lógica de α mediante definiciones, tautologías, teoremas o propiedades, y **refutación** o demostración por contradicción ($\alpha \to \beta \equiv \alpha \wedge \neg\beta \implies \gamma \wedge \neg\gamma$), buscando contraejemplos o encontrando un a tal que $V(\alpha[a] \to \beta[a]) = F$.

Capítulo 3

Problema SAT

3.1. Algoritmos que no requieren cláusulas

3.1.1. Tablas de verdad

Construimos una tabla de verdad por filas, y si en una fila obtenemos que la oración es cierta para dicha interpretación, la oración es satisfacible. Si no es cierta en ninguna, la oración es insatisfacible.

3.1.2. Árboles semánticos

Una hoja de un árbol semántico es un **nodo fallo** si su etiqueta es F, y **nodo éxito** si es V. Representamos el árbol como Υ, y observamos que, para comprobar la satisfacibilidad, basta con encontrar un nodo éxito, y entonces no es necesario seguir desarrollando.

3.1.3. Tableaux semánticos

Un **tableaux semántico** es un árbol en el cual cada nodo está formado por una lista de oraciones (sin paréntesis de ningún tipo), y su raíz es la lista formada por la oración ϕ a desarrollar. Una oración puede tener **comportamiento conjuntivo** (se le puede aplicar una α-fórmula) o **disyuntivo** (se le puede aplicar una β-fórmula), o ser un literal. Para cada rama del árbol:

1. Seleccionamos una oración que no sea un literal, preferiblemente con comportamiento conjuntivo.

2. Si no encontramos ninguna, el nodo es un nodo hoja. Lo marcamos como **cerrado** si contiene un literal y su contrario, de lo contrario como **abierto**.

3. Si tiene comportamiento conjuntivo, aplicamos la α-fórmula correspondiente y dibujamos una rama con la etiqueta α : (nom. de fórmula) y el nodo resultado de sustituir $\alpha, \phi_1, \ldots, \phi_n$ por $\alpha_1, \alpha_2, \phi_1, \ldots, \phi_n$.

4. Si tiene comportamiento disyuntivo, aplicamos la β-fórmula correspondiente y dibujamos dos ramas. La división se marca con la etiqueta β : (nom. de fórmula) y los nodos son los resultantes de sustituir $\beta, \phi_1, \ldots, \phi_n$ por $\beta_1, \phi_1, \ldots, \phi_n$ y $\beta_2, \phi_1, \ldots, \phi_n$.

α-fórmulas			β-fórmulas		
α	α_1	α_2	β	β_1	β_2
$\neg\neg\alpha$	α				
$\alpha \wedge \beta$	α	β	$\neg(\alpha \wedge \beta)$	$\neg\alpha$	$\neg\beta$
$\neg(\alpha \vee \beta)$	$\neg\alpha$	$\neg\beta$	$\alpha \vee \beta$	α	β
$\neg(\alpha \to \beta)$	α	$\neg\beta$	$\alpha \to \beta$	$\neg\alpha$	β
$\alpha \leftrightarrow \beta$	$\alpha \to \beta$	$\beta \to \alpha$	$\neg(\alpha \leftrightarrow \beta)$	$\neg(\alpha \to \beta)$	$\neg(\beta \to \alpha)$

Un **tableaux completado** o **completo** es aquel cuya construcción ha terminado. Decimos que es **cerrado** cuando todas las hojas son cerradas, y **abierto** cuando hay alguna abierta. Así, ϕ es satisfacible si y sólo si su tableau completado es abierto. Este método no detecta tautologías, pero podemos determinar que ϕ es tautológica cuando $\neg\phi$ es insatisfacible.

3.2. Algoritmos que requieren cláusulas

Una fórmula ξ está en **forma normal conjuntiva** si es una cláusula o conjunción de cláusulas. Una **cláusula** es un literal o disyunción de dos o más literales (quitando todos los paréntesis).

- **Cláusula unitaria:** Con un solo literal.
- **Cláusula vacía:** Sin literales. Se denota por $\square$ y es insatisfacible.
- **Cláusula de Horn:** Aquella que tiene como máximo un literal positivo.

Llamamos **conjunto clausal** o **clausulado** al conjunto de dichas cláusulas, y decimos que está en **forma clausal**.

3.2.1. Obtención de FNC

1. Aplicar las reglas de eliminación de $\leftrightarrow$ y $\to$ hasta tener solo $\neg$, $\wedge$ y $\vee$.
2. Aplicar De Morgan hasta que las negaciones solo afecten a átomos.
3. Eliminar las negaciones múltiples ($\neg\neg$).
4. Aplicar distributividad de $\vee$ sobre $\wedge$.
5. Reducir la cantidad de paréntesis.

Si queremos simplificar:

1. Eliminar literales opuestos: $\ell \vee \neg\ell \equiv V$; $\ell \wedge \neg\ell \equiv F$.
2. Eliminar constantes y expresiones repetidas: $V \vee \alpha \equiv V$; $F \vee \alpha \equiv V \wedge \alpha \equiv \alpha$; $F \wedge \alpha \equiv F$; $\alpha \vee \alpha \equiv \alpha \wedge \alpha \equiv \alpha$.
3. Quedarnos con expresiones subsumidas: $(\alpha \vee \beta) \wedge \alpha \equiv \alpha$.

3.2.2. Algoritmo DPLL

Al propagar un literal en una expresión en FNC, lo que hacemos es eliminar las cláusulas en las que aparezca (**cláusula cancelada**) y las ocurrencias de literales complementarios (**ocurrencia eliminada**). Al aplicar DPLL, representamos el conjunto clausulado como un conjunto de conjuntos. Por ejemplo, si $\phi \equiv p \wedge (q \vee \neg r)$, entonces su conjunto clausal es $\Omega_\phi = \{p, q \vee \neg r\}$, y lo representamos como $\Omega_{\phi FNC} = \{p, \{q, \neg r\}\}$. Entonces consideramos 5 reglas:

1. **Regla de la cláusula unitaria:** Si hay una cláusula unitaria, propagar su literal.
2. **Regla del literal puro:** Si hay un literal para el que no se da su complementario, propagarlo.
3. **Regla de la tautología:** Eliminar las cláusulas que contengan literales complementarios.
4. **Regla de la inclusión:** Si existen conjuntos clausales $C_1, C_2 \in \Omega_\phi$ tales que $C_1 \subseteq C_2$, eliminar C_2 de Ω_ϕ.
5. **Regla de ramificación:** Considerar un literal l (normalmente el que aparece más veces) y propagar l por un lado y $\neg l$ por otro.

El algoritmo consiste en:

1. Si $\Omega_\phi = \{\}$, devolver `true` (se indica mediante una flecha de la expresión derivada a la original etiquetada «true»), indicando que es satisfacible.
2. Si $\square \in \Omega_\phi$, devolver `false`.
3. En caso contrario, aplicar la primera de las reglas que sea aplicable y devolver su valor. Para la regla de ramificación: Si la primera rama devuelve `true`, devolverlo. Si no, proceder con la segunda rama y devolver el valor devuelto. Esto es lo que se denomina «backtracking» (a partir de un algoritmo recursivo).

Lo representamos con un grafo, similar al del árbol semántico.

3.2.3. Método de resolución

La **resolvente** de dos cláusulas $\psi \equiv l_1 \vee \cdots \vee l_n \vee j$ y $\phi \equiv k_1 \vee \cdots \vee k_m \vee \neg j$ es la cláusula $R_j(\psi, \phi) \equiv l_1 \vee \cdots \vee l_n \vee k_1 \vee \cdots \vee k_m$. ψ y ϕ son las **cláusulas padres** de $R_j(\psi, \phi)$, y $\{\psi, \phi\}$ es satisfacible si y sólo si $R_j(\psi, \phi)$ lo es. Para resolver por resolución:

1. Definimos el conjunto $\mathcal{C} = \Omega_\phi$ y $\mathcal{C}_0^* = \mathcal{C}$.
2. Por cada par de cláusulas $C_1 = l \vee \alpha$ y $C_2 = \neg l \vee \beta$ en $\mathcal{C}^*$, obtenemos su resolvente y la añadimos a $\mathcal{C}^*$. Si obtenemos la resolvente $\square$, el conjunto $\mathcal{C}^*$ es insatisfacible y por tanto $\mathcal{C}$ también.
3. Una vez obtenidas todas las resolventes, si $\square \notin \mathcal{C}^*$, entonces $\mathcal{C}^*$ es satisfacible y por tanto $\mathcal{C}$ también (solo en L0).

Podemos expresar este algoritmo mediante un **grafo de resolución**, en el que de cada par de cláusulas posible parte una línea hacia abajo hacia su resolvente y tenemos cuidado de no incluir un resolvente igual a una cláusula ya existente.

En dicho grafo, las premisas se dice que están en el nivel 0 ($\mathcal{C}$), y si C_1 y C_2 están en los niveles x_1 y x_2, su resolvente está en el nivel máx$\{x_1, x_2\} + 1$, de forma que cada nivel se representa en una misma línea horizontal, y los resolventes se numeran empezando por el nivel más alto (el nivel 0).

Un grafo de resolución es **básico** si solo muestra dos cláusulas y su resolvente; **completo** si contiene la unión de todos los grafos de resolución básicos, y es un **grafo de refutación** si aparece $\square$.

Para la elección de los pares de cláusulas padres, existen principalmente dos **estrategias de resolución:**

- **Búsqueda en anchura:** Se obtienen todas las resolventes de un nivel antes de pasar al siguiente. Orden $(2,1)$, $(3,1)$, $(3,2)$, $(4,1)$, etc.
- **Búsqueda en profundidad:** Una vez obtenida una resolvente de nivel i, intenta obtener una de nivel $i+1$ a partir de ella. Orden $(2,1) \to i$, $(i,1)$, $(i,2)$, $(i,3) \to j$, $(j,1)$, etc. No es completa.

En la práctica se utiliza la **notación** o **representación Fitting**: Se crea una lista numerada de las cláusulas de $\mathcal{C}$, con la indicación «Premisa» o «C.E.» (conjunto de entrada) a la derecha de cada una. Después, se van comprobando los pares de cláusulas en la lista, empezando por $(2,1)$, $(3,1)$, $(3,2)$, etc., y se va ampliando con los resolventes, que tendrán la indicación $R_l(i,j)$, siendo i y j los ordinales de las cláusulas padres. Para optimizar, se pueden tachar cláusulas por los siguientes motivos:

- Literal puro (l) [después de tachar (n)].
- Subsumida en (n).
- Tautología $(l \vee \neg l)$.

El motivo debe indicarse a la derecha. Si al final queda $\mathcal{C}^* = \{\}$, la oración es satisfacible.

3.3. Razonamiento automático

Un razonamiento es válido ($\{\alpha_1, \ldots, \alpha_n\} \vDash \beta$) cuando $\vDash \alpha_1 \wedge \cdots \wedge \alpha_n \to \beta$, es decir, cuando $\alpha_1 \wedge \cdots \wedge \alpha_n \wedge \neg\beta$ es insatisfacible. Así, mediante tablas de verdad o árboles semánticos, podemos determinar la validez de un razonamiento demostrando que $\alpha_1 \wedge \cdots \wedge \alpha_n \to \beta$ es una tautología. Para el resto de los métodos, comprobamos que $\alpha_1 \wedge \cdots \wedge \alpha_n \wedge \neg\beta$ es una contradicción.

En el caso de demostración por resolución, las cláusulas de $\neg\beta$ se denominan «conjunto soporte de entrada». Se marcan con un $*$ a la izquierda del ordinal tanto estas como las resolventes de alguna cláusula marcada con $*$, de forma recursiva.

Capítulo 4

Deducción natural

Las expresiones, numeradas, están dentro de una caja, que se representa con los tres bordes de arriba, abajo y a la izquierda (a la derecha del número de línea). La caja principal no se suele representar. Si se usa una *Hipótesis*, se crea una nueva caja (dentro de otra), y en cada caja las *Premisas* o *Hipótesis* van al principio y la última línea es la conclusión. Cada caja representa una deducción ($\vdash$).

4.1. Reglas

	Eliminación de	Introducción de
la conjunción	$E_\wedge : \frac{\vdash \alpha \wedge \beta}{\vdash \alpha}, \frac{\vdash \alpha \wedge \beta}{\vdash \beta}$	$I_\wedge : \frac{\vdash \alpha \vdash \beta}{\vdash \alpha \wedge \beta}$
la disyunción	$E_\vee : \frac{\vdash \alpha \vee \beta \vdash (\alpha \vdash \gamma) \vdash (\beta \vdash \gamma)}{\vdash \gamma}$ Prueba por casos	$I_\vee : \frac{\vdash \alpha}{\vdash \alpha \vee \beta}, \frac{\vdash \alpha}{\vdash \beta \vee \alpha}$
la implicación	$E_\rightarrow : \frac{\vdash \alpha \rightarrow \beta \vdash \alpha}{\vdash \beta}$ Modus Ponens	$I_\rightarrow : \frac{\vdash (\alpha \vdash \beta)}{\vdash \alpha \rightarrow \beta}$ Teorema de la Deducción
la doble implicación	$E_\leftrightarrow : \frac{\vdash (\alpha \leftrightarrow \beta)}{\vdash (\alpha \rightarrow \beta)}, \frac{\vdash (\alpha \leftrightarrow \beta)}{\vdash (\beta \rightarrow \alpha)}$	$I_\leftrightarrow : \frac{\vdash (\alpha \rightarrow \beta) \vdash (\beta \rightarrow \alpha)}{\vdash (\alpha \leftrightarrow \beta)}$
la negación	$E_\neg : \frac{\vdash (\neg \alpha \vdash \beta \wedge \neg \beta)}{\vdash \alpha}$ Reducción al absurdo	$I_\neg : \frac{\vdash (\alpha \vdash \beta \wedge \neg \beta)}{\vdash \neg \alpha}$ Reducción al absurdo
la doble negación	$E_{\neg\neg} : \frac{\vdash \neg\neg \alpha}{\vdash \alpha}$	
la contradicción	$CONTRA : \frac{\vdash \alpha \vdash \neg \alpha}{\vdash \beta}$	

También la iteración: $IT : \frac{\vdash \alpha}{\vdash \alpha}$, que se usa cuando la hipótesis de una caja es la conclusión, dado que deben estar en líneas distintas.

Capítulo 5

Lógica categórica

5.1. Conjuntos

Una **categoría** o **conjunto** es una colección no ordenada de **elementos**. Se dice que un elemento x pertenece al conjunto C, y se representa como $x \in C$ (negación: $x \notin C$) o $C(x)$. Podemos definir un conjunto por extensión ($C = \{x_1, x_2, \dots\}$), intensión ($C = \{x|P(x)\}$) o recursión ($C = \{x|R(x)\}$).

- **Igualdad:** $A = B :\iff (x \in A \iff x \in B)$.
- **Inclusión:** $A \subseteq B :\iff (x \in A \implies x \in B)$ (negación: $A \nsubseteq B$).
- **Inclusión estricta:** $A \subsetneq B :\iff (A \subseteq B \text{ y } A \neq B)$.
- **Conjunto total** o **universo:** $\mathcal{U}$, el mayor conjunto que podemos considerar para un estudio.
- **Conjunto vacío:** $\emptyset = \{\}$, sin elementos.
- **Partes:** $\mathcal{P}(X) = \{A|A \subseteq X\}$.
- **Unión:** $A \cup B \coloneqq \{x|x \in A \text{ ó } x \in B\}$.
- **Intersección:** $A \cap B \coloneqq \{x|x \in A \text{ y } x \in B\}$. Si $A \cap B = \emptyset$, A y B son **disjuntos**.
- **Diferencia:** $A - B \coloneqq A \backslash B \coloneqq \{x|x \in A \text{ y } x \notin B\}$
- **Complemento:** $\overline{A} \coloneqq A^{\complement} \coloneqq \mathcal{U} \backslash A$.

Un **diagrama de Euler** representa los conjuntos como círculos bien unos dentro de otros, separados o intersecados, indicando de esta forma sus relaciones. Un **diagrama de Venn** representa los conjuntos como círculos todos intersecados entre sí, con las partes no vacías sombreadas.

Una **familia de conjuntos** $\mathcal{A}$ es un conjunto formado solo por conjuntos, y es una **partición** de A si y sólo si $\bigcup \mathcal{A} = A$ y si para todo $B, C \in \mathcal{A}$ con $B \neq C$ se tiene que $B \cap C = \emptyset$.

5.2. Sintaxis

Extensión de la lógica proposicional. Las proposiciones atómicas tienen la forma $P(x)$, donde P es una categoría y x una variable (ambas conjuntos de letras latinas), y se leen «x es P». Además, se añaden los cuantificadores $\forall x$ (para todo x) y $\exists x$ (existe x), donde x puede ser cualquier variable. Estos tienen la misma prioridad que la negación. Las proposiciones compuestas se forman mediante cuatro **formas normales:**

- **Universal afirmativa:** $\forall x(P(x) \to Q(x))$; «todo P es Q».
- **Universal negativa:** $\forall x(P(x) \to \neg Q(x))$; «ningún P es Q».
- **Existencial afirmativa:** $\exists x(P(x) \wedge Q(x))$; «algún P es Q».
- **Existencial negativa:** $\exists x(P(x) \wedge \neg Q(x))$; «algún P no es Q».

5.3. Evaluación

Para evaluar una proposición en LC interpretada en un mundo $\mathcal{M}$:

1. Definimos $\mathcal{U}$ como el conjunto de todos los elementos que aparecen en $\mathcal{M}$.
2. Identificamos cada categoría P con un conjunto $P_{\mathcal{M}}$ del mundo. El resultado es la **interpretación** $I = \{P \mapsto P_{\mathcal{M}}, \dots\}$.
3. Evaluamos el valor de verdad de la proposición a partir de la interpretación. Para ello:
   a) Si encontramos un $\forall x\alpha[x]$, decimos que esto es verdad si para cualquier $x \looparrowright d$ se cumple $V(\alpha[d]) = V$. Aquí, $\alpha[d] \equiv \{\alpha[x]\}_{d/x}$ el resultado de aplicar la **sustitución** $\{d/x\}$. Entonces comprobamos $V(\alpha[d])$ para todo $x \looparrowright d$ (**asignación**) con $d \in \mathcal{U}$ hasta encontrar un caso donde $V(\alpha[d]) = F$ (con lo que $V(\forall x\alpha[x]) = F$) o llegar a que en todos $V(\alpha[d]) = V$ (con lo que $V(\forall x\alpha[x]) = V$).
   b) Si encontramos un $\exists x\alpha[x]$ decimos que esto es verdad si encontramos un $x \looparrowright d$ para el que $V(\alpha[d]) = V$. Entonces comprobamos $V(\alpha[d])$ para todo $x \looparrowright d$ con $d \in \mathcal{U}$ hasta encontrar un caso donde $V(\alpha[d]) = V$ (con lo que $V(\exists x\alpha[x]) = V$) o llegar a que todos son falsos (con lo que $V(\exists x\alpha[x]) = F$).

Capítulo 6

Lógica de predicados de primer orden

6.1. Relaciones

La lógica de primer orden (L1) extiende la lógica categórica permitiendo expresar relaciones fuera de las formas normales y relaciones de varios objetos. Podemos distinguir:

- **Categorías:** «x es P», $x \in P$, $P(r)$.
- **Relaciones binarias:** «x e y son R», «x se relaciona con y», $(x, y) \in R$, $R(x, y)$, xRy.
- **Relaciones** de cualquier orden: $(x_1, \ldots, x_n) \in S$, $S(x_1, \ldots, x_n)$. Se dice que S tiene **aridad** n, o que es una relación n**-aria**, lo que se representa por S/n. En general, Q es una relación n-aria entre $A_1, \ldots, A_n$ si $Q \subseteq \prod_{i=1}^{n} A_i$. Si $\forall i, A_i = A$, entonces $\prod_{i=1}^{n} A_i = A^n$.

En relaciones con aridad $n \geq 2$, se define el **dominio** como $\mathrm{Dom}(R) = \{(x_1, \ldots, x_{n-1}) | \exists x_n \mid (x_1, \ldots, x_n) \in R\}$ (si la aridad es 2, entonces $\mathrm{Dom}(R) = \{x | \exists y \mid xRy\}$), y el **rango** como $\mathrm{Ran}(R) = \{x_n | \exists (x_1, \ldots, x_{n-1}) \mid (x_1, \ldots, x_n) \in R\}$ (si la aridad es 2, entonces $\mathrm{Ran}(R) = \{y | \exists x \mid xRy\}$. El **campo** de R se define como $\mathrm{Campo}(R) = \mathrm{Dom}(R) \cup \mathrm{Ran}(R)$. Representaciones:

- **Cartesiana:** Similar a una función, con el conjunto inicial en el eje horizontal. Se marcan los puntos que están en R.
- **Tabular:** Como la cartesiana pero en una tabla. En cada celda se pone un 1 si el producto de tipos está en R, un 0 si no está y se deja en blanco si no lo sabemos.
- **Mediante digrafo:** Se representa a la izquierda el conjunto inicial y a la derecha el final, y las relaciones se representan con flechas entre elementos de cada.
- **Grafo dirigido:** Se representa $\mathcal{U}$ y se indican las relaciones binarias con flechas.

Una relación n-aria f es una **función** si y sólo si para cada elemento $x \in \mathrm{Dom}(f)$ existe un único $y \in \mathrm{Ran}(f)$ que se relacione con él. Se escribe $f(x) = y$, y la función se representa como $f : \prod_{i=1}^{n-1} A_i \to A_n$. Es **inyectiva** si $f(x) = f(x') \implies x = x'$, **suprayectiva** si $\mathrm{Ran}(f) = A_n$ y **biyectiva** si es inyectiva y suprayectiva. Definimos la aridad de f como función como $n - 1$.

6.2. Sintaxis

- **Proposición atómica:** V, F o un predicado.
- **Predicado:** Secuencia de letras latinas que representa una relación, seguida de una serie de términos: $R(t_1, \ldots, t_n)$.
- **Término:** Constante que representa un objeto definido, variable o función.
- **Constante:** Secuencia de letras latinas que representa a un objeto definido (salvo V y F).
- **Variable:** Secuencia de letras latinas que representa a un objeto indefinido. Puede estar **ligada** a un cuantificador, y entonces es igual al resto de variables ligadas al mismo, o **libre**, en cuyo caso puede representar cualquier cosa.
- **Función:** Secuencia de letras latinas que representa una función, seguida de una serie de términos: $f(t_1, \ldots, t_n)$.

La construcción de f.b.f es igual que en L0, pero cambiando la forma de las proposiciones atómicas y añadiendo que si α es f.b.f. también lo son $(\forall x \alpha)$ y $(\exists x \alpha)$. Una f.b.f. es **cerrada** si todas las variables están ligadas y **abierta** en otro caso.

6.3. Interpretación y asignación

Una **interpretación** de α en un **mundo** $\mathbb{M}$ es una cuaterna $\mathcal{I}_\alpha = (\mathbb{D}, \mathcal{C}_\mathbb{D}, \mathcal{F}_\mathbb{D}, \mathcal{R}_\mathbb{D})$ donde $\mathbb{D}$ es un conjunto no vacío de objetos, llamado dominio, $\mathcal{C}_\mathbb{D}$ es un conjunto de objetos concretos $(\mathcal{C}_\alpha \mapsto \mathcal{C}_\mathbb{D})$, $\mathcal{F}_\mathbb{D}$ de funciones concretas $(f_\alpha \mapsto f_\mathbb{D})$ y $\mathcal{R}_\mathbb{D}$ de relaciones concretas $(R_\alpha \mapsto R_\mathbb{D})$. La **signatura** es el conjunto de todos los predicados y funciones, indicando su aridad.

Una asignación de variables es una función $\sigma_{\mathcal{I}_\alpha} : \mathcal{V} \to \mathbb{D}$ que relaciona cada variable de α con un elemento del dominio, y definimos $\sigma_{\mathcal{I}_\alpha | x \mapsto d}$ a la asignación definida igual que $\sigma_{\mathcal{I}_\alpha}$ pero asignando a x el objeto d.

Una **asignación de valores de verdad** $v_{\sigma_{\mathcal{I}_\alpha}} : \mathcal{P}_\alpha \to \mathbb{B}$ asigna un valor de verdad a cada elemento atómico de α. Así, $v(R_\alpha(t_1, \ldots, t_n)) = V \iff (d_1, \ldots, d_n) \in R_\mathbb{D}$, donde si t_i es constante entonces $d_i = t_i$, si $t_i = f(x_1, \ldots, x_n)$ entonces d_i es el único y tal que $(x_1, \ldots, x_n, y) \in f$, y si es variable entonces depende de la **asignación**.

La **evaluación** de una oración α se hace igual que en LC, pero partiendo de esta asignación de valores de verdad. También se puede hacer mediante tablas de verdad, que en L1 sólo evalúan una interpretación a la vez:

1. Se introduce una columna por variable, dividida en una fila por cada valor del dominio. Puede ser necesario considerar aquí todas las posibles combinaciones de variables.
2. Se introduce una columna por cada función que aparece en la oración, y se evalúa de acuerdo al valor de la variable dado.
3. Se introducen las filas correspondientes a la fórmula, indicando el orden de evaluación. Un cuantificador que no está dentro de otro ocupa la fila completa, pero su contenido se divide en una fila por cada posible asignación de la variable. Una vez se conoce el valor del

cuantificador no es necesario evaluar el resto de asignaciones, pero es importante justificar los valores de verdad de los predicados (ejemplos: $V : (a, b) \in P_{\mathcal{M}}$; $F : (c, a) \notin Q_{\mathcal{M}}$).

Este método es impráctico, por lo que no se usa.

6.4. Sustituciones

Una **sustitución** es una expresión $s = \{t_1/v_1, \ldots, t_n/v_n\}$ que indica que toda ocurrencia de cada v_i se debe sustituir por el término t_i. Todas las sustituciones se hacen simultáneamente.

Una **particularización por sustitución** consiste en sustituir sus variables por términos. Escribimos Ps como la particularización de la expresión P según la sustitución s.

- En una **particularización básica**, los términos son constantes.
- En una **particularización alfabética**, los términos son otras variables.

Composición de sustituciones: Dadas $s = \{a_1/x_1, \ldots, a_n/x_n\}$ y $t = \{b_1/y_1, \ldots, b_m/y_m\}$ con X e Y los conjuntos de variables sustituidas respectivamente según s y t, $s{\cdot}t = \{(a_i t)/x_i | x_i \neq a_i t\} \cup \{b_i/y_i | y_i \in Y \backslash X\}$, donde $a_i t$ es la particularización de a_i según t.

6.5. Equivalencias

$\neg\exists x\alpha[x] \equiv \forall x\neg\alpha[x]$	$\neg\forall x\alpha[x] \equiv \exists x\neg\alpha[x]$
$\forall x(\alpha[x] \wedge \beta[x]) \equiv \forall x\alpha[x] \wedge \forall x\beta[x]$	$\exists x(\alpha[x] \vee \beta[x]) \equiv \exists x\alpha[x] \vee \exists x\beta[x]$

También, tanto en L1 como en LC, podemos sustituir el nombre de una variable por otro siempre que lo cambiemos en el cuantificador al que está ligado y en todos los símbolos ligados al mismo cuantificador (o bien la variable sea libre), y al hacerlo todas las variables de la oración sigan ligadas al mismo cuantificador de partida (o sigan libres).

6.6. Satisfacibilidad

Podemos comprobar la satisfacibilidad de una oración mediante tableaux. Añadimos dos tipos de reglas:

- **γ-reglas**: $\forall x\alpha[x] \mapsto \alpha[C], \forall x\alpha[x]$; $\neg\exists x\alpha[x] \mapsto \neg\alpha[C], \neg\exists x\alpha[x]$. La sustitución $\{C/x\}$ se hace sobre una constante C existente. Si no existe ninguna, debemos suponer una nueva. El $\forall x\alpha[x]$ resultante no hace referencia a C, de modo que se debe escribir una lista debajo de cada expresión de este tipo ($L = \{\ldots\}$) con los elementos a los que sí hace referencia.
- **δ-reglas**: $\exists x\alpha[x] \mapsto \alpha[C]$; $\neg\forall x\alpha[x] \mapsto \neg\alpha[C]$. La sustitución $\{C/x\}$ se hace sobre una constante C nueva, y entonces se debe añadir a las listas de todas las expresiones de γ-reglas dicha constante.

Al aplicar estas reglas, se debe indicar, por ejemplo: $\gamma : \forall x; \{C/x\}; C$ nueva (la última parte se incluye siempre en las δ-reglas). Las δ-reglas se aplican después de las β-reglas y antes de las γ-reglas, y si se llega a un bucle por una rama, se razona que el tableaux es abierto.

Si el tableaux es cerrado (si todas las hojas están cerradas), llegamos a una contradicción. Sin embargo, si el tableaux es abierto, no sabemos que sea satisfacible (salvo si todos los predicados son de aridad 1 o la identidad). No obstante, los nodos abiertos pueden servir como ejemplos de interpretaciones en las que la oración es satisfacible.

6.7. Grafos semánticos

Son iguales que en L0, pero en los cuantificadores, el nombre de la variable se incluye en el nombre del propio nodo junto con el cuantificador. Además, debajo de cada predicado (que se escribe completo), se puede indicar el f.b.f. de términos, que consiste en añadir un nodo hijo por cada término. Si el término es una función, se indica simplemente el nombre de la función y sus parámetros se escriben como nodos hijo.

6.8. Deducción natural

Se añaden reglas de deducción natural:

- $E_\forall : \frac{\vdash \forall x \alpha[x]}{\vdash \alpha[C]}$. C es una constante cualquiera. Se debe indicar la sustitución $\{C/x\}$ y, en su caso, si C es nueva o «arbitraria».
- $I_\forall : \frac{\vdash \alpha[C]}{\vdash \forall x \alpha[x]}$. C debe ser «arbitraria», es decir, no distinguible de cualquier otro individuo por suposiciones, derivaciones o premisas anteriores. Puede ser obtenida nueva con $E_\forall$.
- $E_\exists : \frac{\vdash \exists x \alpha[x] \vdash (\alpha[C] \vdash \beta)}{\vdash \beta}$. No se debe hacer ninguna suposición sobre C, y β no puede depender de C.
- $I_\exists : \frac{\vdash \alpha[C]}{\vdash \exists x \alpha[x]}$. Se pueden cambiar todas las apariciones de C o solo algunas.

Capítulo 7

Sistemas deductivos, razonamientos y deducciones

Un **sistema deductivo** es un conjunto de axiomas y reglas de inferencia sintácticas ($\vdash$). Una **demostración** o **prueba formal** es una secuencia de conjuntos de fórmulas en las que cada fórmula es un axioma o puede obtenerse del conjunto anterior mediante una regla de inferencia. Cada elemento α del último conjunto de la secuencia se llama **teorema por deducción**, y se dice que α es **demostrable**, lo que escribimos como $\vdash \alpha$.

Un sistema deductivo en L0 y L1 es **sólido** si y sólo si $\vdash \alpha \implies \vDash \alpha$, es decir, si cualquier conclusión **derivable** o **deducible** a partir de las reglas es válida, y es **completo** cuando $\vDash \alpha \implies \vdash \alpha$. Un conjunto de reglas es **inconsistente** si $\vdash \alpha \wedge \neg\alpha$, y es **consistente** si no es inconsistente.

Dada una oración α y un conjunto de oraciones $\mathcal{F}$, $\vDash \alpha$ significa que α es válida y $\mathcal{F} \vDash \alpha$ significa que α es consecuencia lógica de $\mathcal{F}$. Por su parte $\vdash \alpha$ significa que α es demostrable y $\mathcal{F} \vdash \alpha$ significa que α es deducible de $\mathcal{F}$, y representa una **deducción** o **razonamiento**, donde α es la conclusión o **derivación** y las $\psi \in \mathcal{F}$ son las premisas, las fórmulas usadas para llegar a α.

Decimos que un conjunto de oraciones $\mathcal{T}$ es una **teoría** si $\forall\alpha(\mathcal{T} \vDash \alpha \implies \alpha \in \mathcal{T})$, y entonces cada $\alpha \in \mathcal{T}$ es un **teorema**. Una teoría es **axiomatizable** si existe un subconjunto $\mathcal{F}$ tal que $\mathcal{T} = \{\alpha | \mathcal{F} \vDash \alpha\}$, y cada $\alpha \in \mathcal{F}$ es un **axioma**, y es **contradictoria** o **inconsistente** cuando $\mathcal{T} \vDash \alpha$ y $\mathcal{T} \vDash \neg\alpha$.

Una teoría es **decidible** si se puede determinar la consistencia o inconsistencia de una fórmula mediante un algoritmo; **semidecidible** si hay fórmulas cuya inconsistencia no puede ser probada algorítmicamente, e **indecidible** si no es posible crear un algoritmo que determine la consistencia o inconsistencia de una fórmula.

Hilbert opinaba que todo sistema fundamental matemático debía ser consistente, completo y decidible, pero Kurt **Gödel** demostró que ningún sistema capaz de representar los números naturales puede ser a la vez consistente y completo, y que la consistencia no puede probarse con los propios axiomas del sistema, por lo que habrá verdades que no se pueden demostrar. Alan **Turing**, por su parte, demostró que solo sistemas muy restrictivos son decidibles.

Un sistema deductivo cumple el **teorema de la deducción** si verifica que, dado el conjunto $\mathcal{F}$ y las fórmulas α y β, $\mathcal{F} \cup \{\alpha\} \vdash \beta \iff \mathcal{F} \vdash \alpha \to \beta$. Así, $\mathcal{F} \cup \{\alpha_1, \ldots, \alpha_n\} \vdash \beta \iff$

$\mathcal{F} \cup \{\alpha_1, \ldots, \alpha_{n-1}\} \vdash \alpha_n \to \beta \iff \cdots \iff \mathcal{F} \vdash \alpha_1 \to (\alpha_2 \to \cdots (\alpha_n \to \beta) \cdots)$. Este teorema simplifica mucho las demostraciones, si bien no se probó su corrección hasta 1930. No todos los sistemas cumplen en teorema de la deducción, si bien el Teorema de la Deducción Semántica ($\mathcal{F} \cup \{a\} \vDash \beta \iff \mathcal{F} \vDash \alpha \to \beta$) se cumple siempre. Decimos que $\mathcal{F}$ es **sintácticamente completo** si para todo α, $\mathcal{F} \vdash \alpha$ o $\mathcal{F} \vdash \neg\alpha$.

Un **razonamiento deductivo** es el que parte de unas hipótesis básicas para obtener unas consecuencias ($\alpha \vDash \beta$). Normalmente parte de premisas sobre aspectos generales para concluir aspectos particulares. La relación entre premisas y conclusión es de implicación ($\alpha \vDash \beta \iff \vDash \alpha \to \beta$). Algunos tipos de **demostración** deductiva (de $\vDash \alpha \to \beta$):

- **Vacía** de $\alpha \to \beta$: $\forall v_I, v(\alpha) = F$ (no se usa β).
- **Trivial**: $\forall v_I, v(\beta) = V$ (no se usa α).
- **Directa**: Probar que $\alpha \vDash \beta$ usando definiciones o teoremas ya probados, como el Modus Ponens.
- **Por contrarrecíproco**: $\alpha \to \beta \equiv \neg\beta \to \neg\alpha$.
- **Por contradicción**: $\alpha \to \beta \equiv \alpha \wedge \neg\beta \to \gamma \wedge \neg\gamma$.
- **Indirecta**: Si $\alpha \vDash \gamma$ y $\gamma \vDash \beta$ entonces $\alpha \vDash \beta$.

La **refutación por contraejemplo** consiste en buscar a tal que $v(\alpha[a] \to \beta[a]) = F$.

Un **razonamiento inductivo** consiste en obtener reglas generales a partir de casos particulares. Para ello se observan, registran y analizan hechos y se formulan leyes universales a modo de hipótesis o conjeturas, tras lo cual se diseñan experimentos para ver que estas leyes se cumplen.

La **inducción matemática**, sin embargo, se puede probar de forma deductiva, si bien esto requiere lógica de segundo orden. En $\mathbb{N}$, para un $n_0 \in \mathbb{N}$, tenemos:

- **Principio de inducción débil**: $\{P(n_0)\} \cup \bigcup_{n \geq n_0} \{P(n) \to P(n+1)\} \vDash \forall n \geq n_0, P(n)$.
- **Principio de inducción fuerte**: $\{P(n_0)\} \cup \bigcup_{n \geq n_0} \{P(n_0) \wedge \cdots \wedge P(n) \to P(n+1)\} \vDash \forall n \geq n_0, P(n)$.

El **principio de inducción estructural** para **demostración por recursión** es una generalización de la inducción y afirma que, dado un conjunto de elementos definido por recursión con una serie de casos base y reglas de recursión sobre estos, si una propiedad se cumple para cada caso base, y si en cada regla de recursión si la propiedad se cumple para los parámetros de entrada también se cumple para el elemento resultante de aplicarla, entonces esta propiedad la cumplen todos los elementos del conjunto.