LÓGICA PARA INTELIGENCIA ARTIFICIAL

ÁNGEL GONZÁLEZ PRIETO

1. Lógica proposicional

1.1. Fórmulas de la lógica proposicional. Una fórmula de lógica proposicional es una cadena de texto compuesta de los siguiente elementos.

- Literales: $p, q, r, s, P, Q, R, S \ldots$
- Símbolos conectivos:
  - $\wedge$: Conjunción, "y".
  - $\vee$: Disjunción, "o".
  - $\rightarrow$: Implicación, "implica".
  - $\leftrightarrow$: Equivalencia, "si y solo si".
  - $\neg$: Negación, "no" (Unario).
- Paréntesis (si son necesarios para desambiguar).

Observación 1.1. En ocasiones, se introducen dos símbolos más, $\top$ y $\bot$, que se llaman "verdad" y "falsedad" respectivamente. Son auxiliares y no los utilizaremos aquí.

La fórmulas de la lógica proposicional forman un conjunto, **For**, que se construye de forma recursiva como sigue:

- Si a es un literal, entonces $a \in \mathbf{For}$.
- Si $\psi, \varphi \in \mathbf{For}$, entonces
  - $\psi \wedge \varphi \in \mathbf{For}$.
  - $\psi \vee \varphi \in \mathbf{For}$.
  - $\psi \rightarrow \varphi \in \mathbf{For}$.
  - $\psi \leftrightarrow \varphi \in \mathbf{For}$.
  - $\neg\psi \in \mathbf{For}$.

Observación 1.2. Por la definición anterior, si p es un literal, tanto p como $\neg p$ son fórmulas. Este tipo especial de fórmulas se denomina **átomo**, esto es, un literal o la negación de un literal. También, en muchas ocasiones a los literales se les denomina **proposiciones** o **símbolos proposicionales**, lo que justifica el nombre lógica proposicional.

Ejemplo 1.3. Ejemplos de fórmulas:

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

Ejemplo 1.4. No ejemplos de fórmulas:

- $p\wedge$.
- $\rightarrow p$

- $p \neg q$.
- $\neg \wedge p$.
- $p \vee q \;\; r$.

1.1.1. Ejemplos de formalización en lógica proposicional.

Ejemplo 1.5.

- O los hombres han nacido iguales o no son libres.
  - Han nacido iguales: i
  - Son libres: l

$$i \vee \neg l$$

- O está lloviendo y nevando, o está soplando el viento.
  - Está lloviendo: l
  - Está nevando: n
  - Está soplando el viento: v

$$(l \wedge n) \vee v$$

- Supuesto que Pablo se quede, Luis se irá.
  - Pablo se queda: p
  - Luis se queda: l

$$p \rightarrow \neg l$$

- Luis se irá en caso de que Pablo se quede.
  - Pablo se queda: p
  - Luis se queda: l

$$p \rightarrow \neg l$$

- O Crumm es culpable, o él y Moriarty lo son conjuntamente.
  - Cruum es culpable: c
  - Moriarty es culpable: m

$$c \vee (c \wedge m)$$

- O Holmes lleva razón, o Moriarty y Crumm son o ambos culpables o ambos inocentes; y Crumm es culpable.
  - Holmes lleva razón: h
  - Cruum es culpable: c
  - Moriarty es culpable: m

$$(h \vee ((c \vee m) \vee (\neg c \vee \neg m))) \wedge c$$

- No es el caso que, si la luna está hecha de queso verde, entonces los vehículos espaciales no pueden alunizar en ella.
  - La Luna está hecha de queso verde: l
  - Vehículos espaciales no pueden alunizar: v

$$\neg(l \rightarrow v)$$

1.2. Verdad y consecuencia semántica.

Definición 1.6. Una **evaluación de verdad** (también llamada asignación de verdad) es una función

$$v : \mathbf{For} \rightarrow \{0, 1\}$$

tal que

$$v(\psi \wedge \varphi) = \text{mín}(v(\psi), v(\varphi)) = \begin{cases} 1 & \text{si } v(\psi) = 1 \text{ y } v(\varphi) = 1 \\ 0 & \text{si } v(\psi) = 0 \text{ o } v(\varphi) = 0 \end{cases},$$

$$v(\psi \vee \varphi) = \text{máx}(v(\psi), v(\varphi)) = \begin{cases} 1 & \text{si } v(\psi) = 1 \text{ o } v(\varphi) = 1 \\ 0 & \text{si } v(\psi) = 0 \text{ y } v(\varphi) = 0 \end{cases},$$

$$v(\psi \to \varphi) = \begin{cases} 1 & \text{si } v(\psi) = 0 \text{ o } (v(\psi) = 1 \text{ y } v(\varphi) = 1) \\ 0 & \text{si } v(\psi) = 1 \text{ y } v(\varphi) = 0 \end{cases},$$

$$v(\psi \leftrightarrow \varphi) = \begin{cases} 1 & \text{si } v(\psi) = v(\varphi) \\ 0 & \text{si si } v(\psi) \neq v(\varphi) \end{cases},$$

$$v(\neg\psi) = 1 - v(\psi).$$

Observación 1.7. Debido a las reglas de construcción, el único valor de verdad que no está determinado son los valores de los literales (las proposiciones atómicas). Por ello, v queda completamente caracterizada por la veracidad/falsedad de los literales.

Ejemplo 1.8. Tomamos v con $v(p) = 1$, $v(q) = 0$.

- $v(\neg p \wedge q) = \text{mín}(v(\neg p), v(q)) = \text{mín}(1 - v(p), v(q)) = \text{mín}(1 - 1, 0) = \text{mín}(0, 0) = 0$.
- $v(\neg p \to q)$. Tenemos que $v(\neg p) = 1 - v(p) = 0$ y $v(q) = 1$, luego $v(\neg p \to q) = 1$.
- $v(p \wedge (\neg p \vee q)) = \text{mín}(v(p), v(\neg p \vee q)) = \text{mín}(v(p), \text{máx}(v(\neg p), v(q))) = \text{mín}(v(p), \text{máx}(1 - v(p), v(q))) = \text{mín}(1, \text{máx}(1 - 1, 0)) = \text{mín}(1, 0) = 0$.

1.2.1. Tablas de verdad. Existe una forma más sencilla de evaluar la veracidad o falsedad de una fórmula compleja mediante tablas de verdad.

Ejemplo 1.9. $p \wedge (\neg p \vee q)$.

p	q	$p \wedge (\neg p \vee q)$	$\neg p \vee q$	$\neg p$
0	0	0	1	1
0	1	0	1	1
1	0	0	0	0
1	1	1	1	0

Ejemplo 1.10. $(p \leftrightarrow q) \wedge (p \vee \neg q)$.

p	q	$(p \leftrightarrow q) \wedge (p \vee \neg q)$	$p \leftrightarrow q$	$p \vee \neg q$	$\neg q$
0	0	1	1	1	1
0	1	0	0	0	0
1	0	0	0	1	1
1	1	1	1	1	0

Ejemplo 1.11. $(p \wedge q) \to (p \vee q)$.

p	q	$(p \wedge q) \to (p \vee q)$	$p \wedge q$	$p \vee q$
0	0	1	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Esto se llamará más adelante una tautología.

Ejemplo 1.12. $(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$.

p	q	r	$(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$	$\neg p \wedge q$	$(p \to q) \wedge \neg r$	$\neg p$	$p \to q$	$\neg r$
0	0	0	1	0	1	1	1	1
0	0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0
1	1	0	1	0	1	0	1	1
1	1	1	0	0	0	0	1	0

Observación 1.13. De hecho, las tablas de verdad se pueden utilizar para sintetizar la propiedades de la definición 1.6.

ψ	φ	$\psi \wedge \varphi$
0	0	0
0	1	0
1	0	0
1	1	1

ψ	φ	$\psi \vee \varphi$
0	0	0
0	1	1
1	0	1
1	1	1

ψ	φ	$\psi \to \varphi$
0	0	1
0	1	1
1	0	0
1	1	1

ψ	φ	$\psi \leftrightarrow \varphi$
0	0	1
0	1	0
1	0	0
1	1	1

ψ	$\neg\psi$
0	1
1	0

Definición 1.14. Un conjunto de fórmulas $T \subseteq$ **For** se dice **satisfacible** si existe una evaluación de verdad v tal que $v(\psi) = 1$ para todo $\psi \in T$.

Uno de los problemas más importantes en teoría de la computación consiste en, dado un conjunto $T \subseteq$ **For**, decidir si T es satisfacible o no. Para ser exactos, sea n el número de átomos distintos en T (i.e. el número de literales distintos). El problema de comprobar si T es satisfacible se llama el problema n-SAT. Es conocido (teorema de Cook) que n-SAT para $n \geq 3$ pertenece a un tipo de problemas muy difícil llamado **NP**-completo.

Observación 1.15. Por motivos que veremos más adelante, es típico a los subconjuntos $T \subseteq$ **For** llamarlos **teorías**.

1.2.2. Consecuencia semántica.

Definición 1.16. Sea T un conjunto de fórmulas y $\psi \in$ **For**. Diremos que ψ es **consecuencia semántica** de T, y lo escribimos $T \models \psi$, si para toda evaluación de verdad v tal que $v(\varphi) = 1$ para todo $\varphi \in T$, se tiene que $v(\psi) = 1$.

Observación 1.17.

- T puede ser vacío $T = \emptyset$. En ese caso, se escribe $\models \psi$ y se dice que ψ es una **tautología**. Obsérvese que una tautología significa que $v(\psi) = 1$ para toda evaluación v. Es decir, una tautología es una fórmula cuya tabla de verdad devuelve siempre 1.
- Si $T = \{\varphi\}$, se escribe $\varphi \models \psi$. Obsérvese que esto último es equivalente a que $\varphi \to \psi$ sea una tautología, esto es, $\models \varphi \to \psi$.
- Si $\varphi \models \psi$ y $\psi \models \varphi$, se dice que φ y ψ son **equivalentes** y se denota $\varphi \equiv \psi$. Obsérvese que ser equivalente es lo mismo que $\models \varphi \leftrightarrow \psi$.

Ejemplo 1.18. Para toda fórmula $\psi \in$ **For**, se tiene que $\psi \vee \neg\psi$ es una tautología (llamada el principio del tercero excluso).

ψ	$\neg\psi$	$\psi \vee \neg\psi$
0	1	1
1	0	1

Proposición 1.19. *Se tienen las siguientes equivalencias:*

- *Leyes de De Morgan.*
$$\neg(\psi \vee \varphi) \equiv \neg\psi \wedge \neg\varphi, \qquad \neg(\psi \wedge \varphi) \equiv \neg\psi \vee \neg\varphi,$$
- *Doble negación.*
$$\neg\neg\psi \equiv \psi.$$
- *Significado de la implicación.*
$$\psi \to \varphi \equiv \neg\psi \vee \varphi.$$
- *Significado de la equivalencia.*
$$\psi \leftrightarrow \varphi \equiv (\psi \to \varphi) \wedge (\varphi \to \psi) \equiv (\neg\psi \vee \varphi) \wedge (\neg\varphi \vee \psi).$$
- *Idempotencia.*
$$\psi \vee \psi \equiv \psi, \qquad \psi \wedge \psi \equiv \psi.$$
- *Conmutatividad.*
$$\psi \vee \varphi \equiv \varphi \vee \psi, \qquad \psi \wedge \varphi \equiv \varphi \wedge \psi.$$
- *Distributividad.*
$$\psi \vee (\varphi \wedge \sigma) \equiv (\psi \vee \varphi) \wedge (\psi \vee \sigma), \qquad \psi \wedge (\varphi \vee \sigma) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \sigma).$$
- *Asociatividad.*
$$\psi \vee (\varphi \vee \sigma) \equiv (\psi \vee \varphi) \vee \sigma, \qquad \psi \wedge (\varphi \wedge \sigma) \equiv (\psi \wedge \varphi) \wedge \sigma.$$
- *Cancelación de contradicciones y tautologías.*
$$\psi \vee (\varphi \wedge \neg\varphi) \equiv \psi, \qquad \psi \wedge (\varphi \vee \neg\varphi) \equiv \psi.$$

Demostración. Basta calcular la tabla de verdad de los lados izquierdos y derechos de las equivalencias para comprobar que sus valores de verdad coinciden. □

Observación 1.20. En virtud de las propiedades "significado de la implicación" y "significado de la equivalencia", los símbolos $\to$ y $\leftrightarrow$ son redundantes y pueden ser expresados en términos de otros más básicos. Por este motivo, en muchas ocasiones en la literatura estos se consideran como abreviaturas de otros. De hecho, nosotros ignoraremos $\leftrightarrow$ cuando realicemos deducción sintáctica (Sección 1.3).

Observación 1.21.
- Una fórmula ψ es insatisfacible si y solo si $\neg\psi$ es una tautología.
- Más aún, $T \models \psi$ si y solo si $T \cup \{\neg\psi\}$ es insatisfacible.

1.2.3. Formas estándar de una fórmula.

Definición 1.22. Una fórmula se dice en **forma normal conjuntiva (FNC)** si es una conjunción de disyunciones de átomos, esto es, si es de la forma
$$(p_{1,1} \vee p_{1,2} \vee \ldots \vee p_{1,n}) \wedge (p_{2,1} \vee p_{2,2} \vee \ldots \vee p_{2,n}) \wedge \ldots \wedge (p_{m,1} \vee p_{m,2} \vee \ldots \vee p_{m,n}) = \bigwedge_i \bigvee_j p_{i,j}.$$

Análogamente, una fórmula se dice en **forma normal disyuntiva (FND)** si es una disyunción de conjunciones de átomos, esto es, si es de la forma
$$(p_{1,1} \wedge p_{1,2} \wedge \ldots \wedge p_{1,n}) \vee (p_{2,1} \wedge p_{2,2} \wedge \ldots \wedge p_{2,n}) \vee \ldots \vee (p_{m,1} \wedge p_{m,2} \wedge \ldots \wedge p_{m,n}) = \bigvee_i \bigwedge_j p_{i,j}.$$

Proposición 1.23 (Existencia de FNC y FND)**.** *Toda fórmula es equivalente a otra en FNC (resp. FND).*

Demostración. Usar la tabla de verdad de la fórmula. □

Ejemplo 1.24. Pasar a FND la fórmula $(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$.

p	q	r	$(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$	$\neg p \wedge q$	$\neg p$	$(p \to q) \wedge \neg r$	$p \to q$	$\neg r$
0	**0**	**0**	**1**	0	1	1	1	1
0	0	1	0	0	1	0	1	0
0	**1**	**0**	**1**	1	1	1	1	1
0	**1**	**1**	**1**	1	1	0	1	0
1	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0
1	**1**	**0**	**1**	0	0	1	1	1
1	1	1	0	0	0	0	1	0

Escogemos todas las asignaciones que retornan **verdadero** y ponemos las variables **en su forma original** para que sean ciertas solo cuando tomen los valores de la tabla de verdad.

$$\underbrace{(\neg p \wedge \neg q \wedge \neg r)}_{\text{Fila 1}} \vee \underbrace{(\neg p \wedge q \wedge \neg r)}_{\text{Fila 3}} \vee \underbrace{(\neg p \wedge q \wedge r)}_{\text{Fila 4}} \vee \underbrace{(p \wedge q \wedge \neg r)}_{\text{Fila 7}}$$

Ejemplo 1.25. Pasar a FNC la fórmula $(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$.

p	q	r	$(\neg p \wedge q) \vee ((p \to q) \wedge \neg r)$	$\neg p \wedge q$	$\neg p$	$(p \to q) \wedge \neg r$	$p \to q$	$\neg r$
0	0	0	**1**	0	1	1	1	1
0	**0**	**1**	**0**	0	1	0	1	0
0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	0
1	**0**	**0**	**0**	0	0	0	0	1
1	**0**	**1**	**0**	0	0	0	0	0
1	1	0	1	0	0	1	1	1
1	**1**	**1**	**0**	0	0	0	1	0

Escogemos todas las asignaciones que retornan **falso** y ponemos las variables **en su forma negadas** para que sean falsas siempre que no tomen el valor deseado de la tabla de verdad.

$$\underbrace{(p \vee q \vee \neg r)}_{\text{Fila 2}} \wedge \underbrace{(\neg p \vee q \vee r)}_{\text{Fila 5}} \wedge \underbrace{(\neg p \vee q \vee \neg r)}_{\text{Fila 6}} \wedge \underbrace{(\neg p \vee \neg q \vee \neg r)}_{\text{Fila 8}}$$

Equivalentemente: $\text{FNC}(\varphi) = \neg\text{FND}(\neg\varphi)$.

Observación 1.26. A una disjunción de átomos se le denomina una **cláusula**. Como veremos, las cláusulas son un elemento fundamental para probar consecuencia de forma automática.

1.3. Sistemas de deducción. Dada una teoría T, la comprobación de que $T \models \psi$ para una cierta fórmula ψ se reduce, en virtud de la Observación 1.21 a comprobar si $T \cup \{\neg\psi\}$ es insatisfacible. Como comentamos, este es el llamado problema SAT que es computacionalmente intratable cuando el número de literales es alto (de hecho, si n es el número de literales distintas, requiere computar 2^n filas de la tabla de verdad).

Definición 1.27. Sea $T \subseteq$ **For** una teoría y $\psi \in$ **For**. Dado un sistema de reglas, R, se dice que de T **se deduce** ψ o que ψ es **consecuencia sintáctica** de T, y lo denotaremos $T \rightarrowtail_R \psi$ (o simplemente $T \rightarrowtail \psi$) si existe una regla de R que obtiene ψ de un subconjunto de T. Se dice

que una fórmula ψ es un **teorema** de la teoría T, denotado $T \vdash_R \psi$ (o simplemente $T \vdash \psi$), si existe una colección finita de fórmulas $\psi_1, \psi_2, \ldots, \psi_n = \psi$ tal que $T \cup \{\psi_1, \ldots, \psi_{i-1}\} \rightsquigarrow_R \psi_i$ para todo $1 \leq i \leq n$.

Definición 1.28. Un **sistema de deducción**, $\mathcal{D}$, también llamado una **lógica**, consta de un conjunto de fórmulas $T \subseteq$ **For**, llamados los **axiomas**, así como un conjunto de reglas R, llamadas las **reglas de deducción**.

1.3.1. Deducción natural. Es el sistema de deducción más similar a la forma del razonamiento humano. Opera de forma similar a como lo realiza un individuo cuando intenta extraer consecuencias lógicas en un entorno real.

- **Axiomas**: $\emptyset$.
- **Reglas de deducción**: Tiene 10 reglas de deducción.

($\wedge^{in}$) $\psi, \varphi \rightsquigarrow \psi \wedge \varphi$.
($\wedge_1^{out}$) $\psi \wedge \varphi \rightsquigarrow \psi$.
($\wedge_2^{out}$) $\psi \wedge \varphi \rightsquigarrow \varphi$.
($\vee_1^{in}$) $\psi \rightsquigarrow \psi \vee \varphi$.
($\vee_2^{in}$) $\varphi \rightsquigarrow \psi \vee \varphi$.
($\vee^{out}$) $\psi \to \sigma, \varphi \to \sigma, \psi \vee \varphi \rightsquigarrow \sigma$.
($\neg^{in}$) $\psi \to (\varphi \wedge \neg\varphi) \rightsquigarrow \neg\psi$ (Demostración por contradicción).
($\neg^{out}$) $\neg\neg\psi \rightsquigarrow \psi$ (Doble negación).
($\to^{in}$) Para todo $T \subseteq$ **For**, si $T \cup \{\psi\} \vdash \varphi$ entonces $T \rightsquigarrow \psi \to \varphi$ (Regla fantasiosa).
($\to^{out}$) $\psi \to \varphi, \psi \rightsquigarrow \varphi$ (Modus ponens).

Ejemplo 1.29. $s \wedge (p \vee q), p \to \neg r, q \to \neg r \vdash s \wedge \neg r$.

1. $s \wedge (p \vee q), p \to \neg r, q \to \neg r \rightsquigarrow_{\wedge_2^{out}} p \vee q$.
2. $s \wedge (p \vee q), p \to \neg r, q \to \neg r, p \vee q \rightsquigarrow_{\vee^{out}} \neg r$.
3. $s \wedge (p \vee q), p \to \neg r, q \to \neg r, p \vee q, \neg r \rightsquigarrow_{\wedge_1^{out}} s$.
4. $s \wedge (p \vee q), p \to \neg r, q \to \neg r, p \vee q, \neg r, s \rightsquigarrow_{\wedge^{in}} s \wedge \neg r$.

Como esta forma es muy farragosa y ensucia mucho la notación, lo denotaremos de la siguiente forma.

(1)	$s \wedge (p \vee q)$	premisa
(2)	$p \vee q$	$\wedge_2^{out}(1)$
(3)	$p \to \neg r$	premisa
(4)	$q \to \neg r$	premisa
(5)	$\neg r$	$\vee^{out}(2,3,4)$
(6)	s	$\wedge_1^{out}(1)$
(7)	$s \wedge \neg r$	$\wedge^{in}(5,6)$

Ejemplo 1.30. $p \to q, q \to r, p \vdash r$.

(1)	$p \to q$	premisa
(2)	p	premisa
(3)	q	$\to^{out}$
(4)	$q \to r$	premisa
(5)	r	$\to^{out}$

Ejemplo 1.31. $p \to q, q \to r \vdash p \to r$.

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		p	supuesto
(3)		$p \to q$	premisa
(4)		q	$\to^{out}$ $(3, 2)$
(5)		$q \to r$	premisa
(6)		r	$\to^{out}$ $(5, 4)$
(7)	$p \to r$		Fin regla fantasiosa $\to^{in}$ $(2, 6)$

Observación 1.32. Como vimos en la Observación 1.20, el simbolo de equivalencia $\leftrightarrow$ es redundante, puesto que semánticamente puede reducirse a otros símbolos más simples. En este sentido, podemos formular un sistema de reglas en el que no aparezca $\leftrightarrow$, como el anterior, sin perder potencia expresiva. No obstante, resulta cómodo incluir $\leftrightarrow$ como símbolo del lenguaje puesto que nos permite escribir de forma compacta varios resultados interesantes. En ese caso, si queremos utilizarlo, necesitamos añadir las reglas:

$(\leftrightarrow^{out})$ $\psi \leftrightarrow \varphi \rightsquigarrow (\psi \to \varphi) \wedge (\varphi \to \psi)$.
$(\leftrightarrow^{in})$ $(\psi \to \varphi) \wedge (\varphi \to \psi) \rightsquigarrow \psi \leftrightarrow \varphi$.

Proposición 1.33. *Para todo $\psi, \varphi \in$ **For** se tiene que*

- *$\psi \vdash \varphi$ si y solo si $\vdash \psi \to \varphi$.*
- *$\psi \vdash \varphi$ y $\varphi \vdash \psi$ si y solo si $\vdash \psi \leftrightarrow \varphi$. En este caso, se dicen que ψ y φ son equivalentes sintácticos.*

Demostración. Supongamos que $\psi \vdash \varphi$. Entonces, podemos deducir sintácticamente $\vdash \psi \to \varphi$ del siguiente modo

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		ψ	Supuesto
(3)		φ	$\psi \vdash$
(4)	$\psi \to \varphi$		Fin regla fantasiosa $\to^{in}$ $(2, 3)$

Análogamente, si $\psi \to \varphi$ es un teorema, podemos deducir $\psi \vdash \varphi$ del siguiente modo

(1)	$\psi \to \varphi$	Premisa
(2)	ψ	Premisa
(3)	φ	$\to^{out}$ $(1, 2)$

Para la segunda parte, en la que $\psi \vdash \varphi$ y $\varphi \vdash \psi$ si y solo si $\vdash \psi \leftrightarrow \varphi$, basta con aplicar el resultado anterior dos veces descomponiendo $\leftrightarrow$ mediante las reglas $\leftrightarrow^{in}$ y $\leftrightarrow^{out}$ □

Teorema 1.34 (Principio de substitución)**.** *Sea $\Phi \in$ **For** y sean $\psi, \varphi \in$ **For** dos fórmulas equivalentes sintácticas. Denotemos por $\Phi[\varphi/\psi]$ la fórmula que resulta de substituir en Φ todas las apariciones de la fórmula ψ por φ. Entonces*

$$\vdash \Phi \leftrightarrow \Phi[\varphi/\psi].$$

Demostración. La demostración se realiza por inducción en la complejidad de la fórmula Φ. □

Proposición 1.35 (Algunas reglas derivadas)**.** *Las siguientes reglas son teoremas de deducción natural*

IV: $\vdash \psi \to \psi$ *(Implicación vacua).*
PTE: $\vdash \psi \vee \neg\psi$ *(Principio del tercero excluso).*
PC: $\psi \wedge \neg\psi \vdash \varphi$ *para todo* $\varphi \in$ **For** *(Principio de contradicción).*
MT: $\psi \to \varphi, \neg\varphi \vdash \neg\psi$ *(Modus tollens).*

C: $\psi \vee \varphi, \neg\psi \vdash \varphi$ *(Corte).*

Demostración. .

Prueba de IV			
(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		ψ	Supuesto
(3)		ψ	Copia desde (2)
(4)	$\psi \to \psi$		Fin regla fantasiosa $\to^{in}$ $(2,3)$

Prueba de PTE				
(1)	$\rightsquigarrow$			Inicio regla fantasiosa
(2)		$\neg(\psi \vee \neg\psi)$		Supuesto
(3)		$\rightsquigarrow$		Inicio regla fantasiosa
(4)			ψ	Supuesto
(6)			$\psi \vee \neg\psi$	$\vee_2^{in}(4)$
(7)			$\neg(\psi \vee \neg\psi)$	Copia (2)
(8)			$(\psi \vee \neg\psi) \wedge \neg(\psi \vee \neg\psi)$	$\wedge^{in}(6,7)$
(9)		$\psi \to (\psi \vee \neg\psi) \wedge \neg(\psi \vee \neg\psi)$		Fin regla fantasiosa $\to^{in}$ $(4,8)$
(10)		$\neg\psi$		$\neg^{in}(9)$
(11)		$\rightsquigarrow$		Inicio regla fantasiosa
(12)			$\neg\psi$	Supuesto
(13)			$\psi \vee \neg\psi$	$\vee_1^{in}(12)$
(14)			$\neg(\psi \vee \neg\psi)$	Copia (2)
(15)			$(\psi \vee \neg\psi) \wedge \neg(\psi \vee \neg\psi)$	$\wedge^{in}(13,14)$
(16)		$\neg\psi \to (\psi \vee \neg\psi) \wedge \neg(\psi \vee \neg\psi)$		Fin regla fantasiosa $\to^{in}$ $(12,15)$
(17)		ψ		$\neg^{in}(16)$
(18)		$\psi \wedge \neg\psi$		$\wedge^{in}(10,17)$
(19)	$\neg(\psi \vee \neg\psi) \to \psi \wedge \neg\psi$			Fin regla fantasiosa $\to^{in}$ $(2,18)$
(20)	$\neg\neg(\psi \vee \neg\psi)$			$\neg^{in}(19)$
(21)	$\psi \vee \neg\psi$			$\neg^{out}(20)$

Prueba de PC			
(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		$\neg\varphi$	supuesto
(3)		ψ	premisa
(4)		$\neg\psi$	premisa
(5)		$\psi \wedge \neg\psi$	$\wedge^{in}(3,4)$
(6)	$\neg\varphi \to (\psi \wedge \neg\psi)$		Fin regla fantasiosa $\to^{in}$ $(2,5)$
(7)	φ		$\neg^{in}$

Prueba de MT			
(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		ψ	Supuesto
(3)		$\psi \to \varphi$	Premisa
(4)		φ	$\to^{out}$ $(2,3)$
(5)		$\neg\varphi$	Premisa
(6)		$\varphi \wedge \neg\varphi$	$\wedge^{in}(4,5)$
(7)	$\psi \to (\varphi \wedge \neg\varphi)$		Fin regla fantasiosa $\to^{in}$ $(2,6)$
(8)	$\neg\psi$		$\neg^{in}(7)$

Prueba de C			
(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		ψ	Supuesto
(3)		$\neg\psi$	Premisa
(4)		$\psi \wedge \neg\psi$	$\wedge^{in}(2,3)$
(5)		φ	PC
(6)	$\psi \to \varphi$		Fin regla fantasiosa $\to^{in}$ $(2,14)$
(7)	$\varphi \to \varphi$		IV
(8)	$\psi \vee \varphi$		Premisa
(9)	φ		$\vee^{out}$

□

Corolario 1.36 (Modus tollens generalizado)**.** *Si $\psi \vdash \varphi$ entonces $\neg\varphi \vdash \neg\psi$ (Modus tollens generalizado).*

Demostración. Basta juntar la regla derivada MT con la Proposición 1.33. □

Proposición 1.37 (Equivalencias importantes)**.** *Las siguientes reglas son equivalencias sintácticas*

ADN: $\vdash \psi \leftrightarrow \neg\neg\psi$ *(Anulación de la doble negación).*
dM $\vee$*:* $\vdash \neg(\psi \vee \varphi) \leftrightarrow \neg\psi \wedge \neg\varphi$ *(de Morgan* $\vee$*).*
dM $\wedge$*:* $\vdash \neg(\psi \wedge \varphi) \leftrightarrow \neg\psi \vee \neg\varphi$ *(de Morgan* $\wedge$*).*
CI: $\vdash \psi \vee \varphi \leftrightarrow \neg\psi \to \varphi$ *(Conjunción como implicación).*

Demostración.

$\boxed{ADN}$ Para $\vdash \psi \leftrightarrow \neg\neg\psi$, recuérdese que la regla $\neg^{out}$ dice $\neg\neg\psi \vdash \psi$, luego basta probar $\psi \vdash \neg\neg\psi$. Para ello, tenemos la siguiente deducción.

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		$\neg\psi$	Supuesto
(3)		ψ	Premisa
(4)		$\psi \wedge \neg\psi$	$\wedge^{in}(2,3)$
(5)	$\neg\psi \to \psi \wedge \neg\psi$		Fin regla fantasiosa $\to^{in}$ $(2,4)$
(6)	$\neg\neg\psi$		$\neg^{in}(5)$

$\boxed{dM}$ Para probar $\neg(\psi \vee \varphi) \vdash \neg\psi \wedge \neg\varphi$ tenemos que

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		ψ	Supuesto
(3)		$\psi \vee \varphi$	$\vee_2^{in}(2)$
(4)	$\psi \to \psi \vee \varphi$		Fin regla fantasiosa $\to^{in}(2,3)$
(5)	$\neg(\psi \vee \varphi)$		Premisa
(6)	$\neg\psi$		$\neg^{in}(4,5)$
(7)	$\rightsquigarrow$		Inicio regla fantasiosa
(8)		φ	Supuesto
(9)		$\psi \vee \varphi$	$\vee_1^{in}(8)$
(10)	$\varphi \to \psi \vee \varphi$		Fin regla fantasiosa $\to^{in}(8,9)$
(11)	$\neg\varphi$		$\neg^{in}(10,5)$
(12)	$\neg\psi \wedge \neg\varphi$		$\wedge^{in}(6,11)$

Con respecto a $\neg(\psi \wedge \varphi) \vdash \neg\psi \vee \neg\varphi$, aplicando de nuevo esta regla con $\psi' = \neg\psi$ y $\varphi' = \neg\varphi$, obtenemos

$$\neg(\neg\psi \vee \neg\varphi) \vdash \neg\neg\psi \wedge \neg\neg\varphi \vdash_{\wedge^{out}} \neg\neg\psi, \neg\neg\varphi \vdash_{\neg^{out}} \psi, \varphi \vdash_{\wedge^{in}} \psi \wedge \varphi.$$

De este modo, utilizando el Corolario 1.36 obtenemos $\neg(\psi \wedge \varphi) \vdash \neg\neg(\neg\psi \vee \neg\varphi) \vdash \neg\psi \vee \neg\varphi$, como queríamos demostrar.

Para el recíproco $\neg\psi \wedge \neg\varphi \vdash \neg(\psi \vee \varphi)$ tenemos la siguiente deducción

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		$\psi \vee \varphi$	Supuesto
(3)		$\neg\psi$	Premisa
(4)		φ	$C(2,3)$
(5)		$\neg\varphi$	Premisa
(6)		$\varphi \wedge \neg\varphi$	$\wedge^{in}(4,5)$
(7)	$(\psi \vee \varphi) \to (\varphi \wedge \neg\varphi)$		Fin regla fantasiosa $\to^{in}(2,6)$
(5)	$\neg(\psi \vee \varphi)$		$\neg^{in}(7)$

Finalmente, para probar $\neg\psi \vee \neg\varphi \vdash \neg(\psi \wedge \varphi)$ utilizamos que, como acabamos de probar $\psi \wedge \varphi \vdash \neg\neg\psi \wedge \neg\neg\varphi \vdash \neg(\neg\psi \vee \neg\varphi)$. Utilizando ADN obtenemos $\neg\psi \vee \neg\varphi \vdash \neg\neg(\neg\psi \vee \neg\varphi)$, y el resultado se sigue de aplicar modus tollens generalizado.

$\boxed{CI}$ Para la equivalencia entre conjunción e implicación, en primer lugar $\psi \vee \varphi \vdash \neg\psi \to \varphi$ se sigue de la siguiente derivación.

(1)	$\rightsquigarrow$			Inicio regla fantasiosa
(2)		$\neg\psi$		supuesto
(3)		$\rightsquigarrow$		Inicio regla fantasiosa
(4)			ψ	supuesto
(5)			$\neg\psi$	premisa
(6)			$\psi \wedge \neg\psi$	$\wedge^{in}(4,5)$
(7)			φ	PC
(8)		$\psi \to \varphi$		Fin regla fantasiosa $\to^{in}(4,7)$
(9)		$\rightsquigarrow$		Inicio regla fantasiosa
(10)			φ	supuesto
(11)			φ	premisa
(12)		$\varphi \to \varphi$		Fin regla fantasiosa $\to^{in}(10,11)$
(13)		$\psi \vee \varphi$		premisa
(14)		φ		$\vee^{out}(8,12,13)$
(15)	$\neg\psi \to \varphi$			Fin regla fantasiosa $\to^{in}(2,14)$

Para el recíproco, $\neg\psi \to \varphi \vdash \psi \vee \varphi$, se sigue de la siguiente derivación.

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		$\neg(\psi \vee \varphi)$	Supuesto
(3)		$\neg\psi \wedge \neg\varphi$	dM $\vee$
(4)		$\neg\psi$	$\wedge_1^{out}(3)$
(5)		$\neg\psi \to \varphi$	Premisa
(6)		φ	$\to^{out}(5,4)$
(7)		$\neg\varphi$	$\wedge_2^{out}(3)$
(8)		$\varphi \wedge \neg\varphi$	$\wedge^{in}(6,7)$
(9)	$\neg(\psi \vee \varphi) \to (\varphi \wedge \neg\varphi)$		Fin regla fantasiosa $\to^{in}(2,8)$
(10)	$\neg\neg(\psi \vee \varphi)$		$\neg^{in}(9)$
(11)	$\psi \vee \varphi$		$\neg^{out}(10)$

□

Corolario 1.38. *Para cualquier* $\psi \in$ **For** *se tiene que* $\vdash \psi \leftrightarrow \mathrm{FNC}(\psi)$ *y* $\vdash \psi \leftrightarrow \mathrm{FND}(\psi)$.

1.3.2. Sistema axiomático de Łukasiewicz. Es un sistema de deducción más compacto que la deducción natural. Tiene tres esquemas de axiomas, pero una única regla de deducción (el modus ponnens).

- **Axiomas**: Para cualesquiera $\alpha, \beta, \gamma \in$ **For**, las siguientes fórmulas son axiomas.
  A1 $\alpha \to (\beta \to \alpha)$.
  A2 $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$.
  A3 $(\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$.
- **Reglas de deducción**: Tiene una única regla de deducción, el modus ponnens.

$$\psi \to \varphi, \psi \rightarrowtail \varphi.$$

Ejemplo 1.39. $\vdash_L \psi \to \psi$.

(1)	$\psi \to ((\varphi \to \psi) \to \psi)$	A1$(\alpha = \psi, \beta = \varphi \to \psi)$
(2)	$(\psi \to ((\varphi \to \psi) \to \psi)) \to ((\psi \to (\varphi \to \psi)) \to (\psi \to \psi))$	A2$(\alpha = \psi, \beta = \varphi \to \psi, \gamma = \psi)$
(3)	$(\psi \to (\varphi \to \psi)) \to (\psi \to \psi)$	MP(1,2)
(4)	$\psi \to (\varphi \to \psi)$	A1$(\alpha = \psi, \beta = \varphi)$
(5)	$\psi \to \psi$	MP(3,4)

Ejemplo 1.40. $\psi \to \varphi, \varphi \to \sigma \vdash_L \psi \to \sigma$.

(1)	$(\varphi \to \sigma) \to (\psi \to (\varphi \to \sigma))$	A1($\alpha = \varphi \to \sigma, \beta = \psi$)
(2)	$\varphi \to \sigma$	Premisa
(3)	$\psi \to (\varphi \to \sigma)$	MP(1,2)
(4)	$(\psi \to (\varphi \to \sigma)) \to ((\psi \to \varphi) \to (\psi \to \sigma))$	A2($\alpha = \psi, \beta = \varphi, \gamma = \sigma$)
(5)	$(\psi \to \varphi) \to (\psi \to \sigma)$	MP(4,3)
(6)	$\psi \to \varphi$	Premisa
(7)	$\psi \to \sigma$	MP(5,6)

1.4. Completitud.

Definición 1.41. Sea $\mathcal{D}$ un sistema de deducción.

- Se dice que $\mathcal{D}$ es **correcto** si, para todo $T \subseteq$ **For** y toda $\psi \in$ **For**, $T \vdash \psi$ implica $T \models \psi$.
- Se dice que $\mathcal{D}$ es **completo** si, para todo $T \subseteq$ **For** y toda $\psi \in$ **For**, $T \models \psi$ implica $T \vdash \psi$.

Teorema 1.42. *(Corrección) El sistema de deducción natural es correcto.*

Demostración. Para las 9 reglas de deducción natural, salvo $\to^{in}$, basta comprobar, usando tablas de verdad, que definen tautologías. Para $\wedge^{in}, \wedge_1^{out}, \wedge_2^{out}, \vee_1^{in}, \vee_2^{in}, \neg^{out}$ es inmediato utilizando las tablas de verdad de la Observación 1.13. Para probar que $\to^{out}$ es correcta, veamos que $((\psi \to \varphi) \wedge \psi) \to \varphi$ es una tautología.

ψ	φ	$\psi \to \varphi$	$(\psi \to \varphi) \wedge \psi$	$((\psi \to \varphi) \wedge \psi) \to \varphi$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

Respecto a $\neg^{in}$, veamos que $(\psi \to (\varphi \wedge \neg\varphi)) \to \neg\psi$ es una tautología.

ψ	φ	$\neg\psi$	$\neg\varphi$	$\varphi \wedge \neg\varphi$	$\psi \to (\varphi \wedge \neg\varphi)$	$(\psi \to (\varphi \wedge \neg\varphi)) \to \neg\psi$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	0	1
1	1	0	0	0	0	1

Para $\vee^{out}$ tenemos que $((\psi \to \sigma) \wedge (\varphi \to \sigma) \wedge (\psi \vee \varphi)) \to \sigma$ es una tautología, como se muestra en la siguiente tabla.

ψ	φ	σ	$\psi \to \sigma$	$\varphi \to \sigma$	$\psi \vee \varphi$	$(\psi \to \sigma) \wedge (\varphi \to \sigma) \wedge (\psi \vee \varphi)$	$((\psi \to \sigma) \wedge (\varphi \to \sigma) \wedge (\psi \vee \varphi)) \to \sigma$
0	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1
0	1	0	1	0	1	0	1
0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1
1	0	1	1	1	1	1	1
1	1	0	0	0	1	0	1
1	1	1	1	1	1	1	1

Finalmente, respecto a la regla fantasiosa $\to^{in}$, podemos argumentar por inducción en el número de veces que es aplicada. Si no se ha aplicado en ninguna ocasión, la comprobación de veracidad se reduce a los cálculos anteriores. Por otro lado, si $\psi \vdash \varphi$, por hipótesis de inducción esta deducción sintáctica es correcta y, por tanto $\psi \models \varphi$ o, equivalentemente, $\models \psi \to \varphi$, lo que finaliza la prueba. □

Nos dirigimos ahora a probar algo mucho más fuerte: que el sistema de deducción natural es completo. Para ello, puede parecer necesario probar que todas las implicaciones semánticas son demostrables sintácticamente. Sin embargo, gracias al siguiente resultado, podemos reducirnos únicamente a las tautologías.

Teorema 1.43 (Compacidad)**.** *Sea $T \subseteq$ **For** una teoría (posiblemente infinita) y $\psi \in$ **For**. Si $T \models \psi$, entonces existe un subconjunto finito $T_0 \subseteq T$ tal que $T_0 \models \psi$.*

Esquema de la demostración. Supongamos que ψ tiene n literales distintos. Para cada una de las posibles asignaciones de verdad v para estos n literales tales que $v(\psi) = 0$, tiene que existir $\varphi_v \in T$ tal que $v(\varphi_v) = 0$ (de lo contrario, $v(\varphi) = 1$ para todo $\varphi \in T$, contradiciendo que $T \models \psi$). De este modo, basta tomar $T_0 = \{\varphi_v \,|\, v(\psi) = 0\}$. □

Observación 1.44. La prueba anterior no es completamente correcta. Podría ocurrir que la fórmula φ_v tuviese más de n literales, por lo sería necesario considerar más y más asignaciones de verdad de modo que T_0 fuese infinito. La prueba precisa de este resultado es mucho más complicada y se conoce como **teorema de compacidad**. En su forma más general, este resultado dice que, si para una teoría T se tiene que todo subconjunto finito $T_0 \subseteq T$ es satisfacible, entonces T es satisfacible.

Proposición 1.45. *Un sistema de deducción $\mathcal{D}$ es completo si y solo sí todas sus tautologías son demostrables i.e. si $\models \psi$ implica $\vdash \psi$ para toda $\psi \in$ **For**.*

Demostración. Es obvio que si un sistema es completo entonces todas sus tautologías son demostrables. Consideremos el recíproco, esto es, que en $\mathcal{D}$ todas las tautologías son demostrables. Sea $T \subseteq$ **For** una teoría y supongamos que $T \models \psi$. Por el Teorema 1.43 tenemos que $T_0 \models \psi$ para cierto subconjunto finito $T_0 \subseteq T$. Ahora bien, como T_0 es finito, podemos formar la fórmula $\bigwedge_{\varphi \in T_0} \varphi$. De este modo, tenemos que $T_0 \models \psi$ si y solo si $\models \left(\bigwedge_{\varphi \in T_0} \varphi \right) \to \psi$. Como las tautologías son demostrables, tenemos que $\vdash \left(\bigwedge_{\varphi \in T_0} \varphi \right) \to \psi$ y, aplicando $\to^{out}$, se tiene que $T_0 \vdash \psi$. En particular, esto significa que $T \vdash \psi$, como queríamos demostrar. □

Observación 1.46. En lógicas más generales, puede ocurrir que un sistema de deducción sea capaz de probar todas sus tautologías, pero no sea capaz de probar todos sus teoremas. Cuando un sistema cumple que todas sus tautologías son demostrables (i.e. $\models \psi$ implica $\vdash \psi$) se dice que es **débilmente completo**. El resultado anterior dice que, en lógica proposicional, ser débilmente completo es equivalente a ser completo.

Gracias al resultado anterior, podemos centrar nuestra atención en las tautologías de la lógica proposicional. Para ello, probemos un resultado sobre la estructura de su FND.

Lema 1.47. *Sea $\psi \in$ **For** una tautología semántica (i.e. $\models \psi$). Supongamos que ψ está formado por los átomos $p_1, \ldots, p_n$. Escribamos la Forma Normal Conjuntiva de ψ como*

$$\psi = \bigwedge_{i=1}^{m} C_i,$$

donde las C_i son cláusulas (i.e. conjunciones de átomos). Entonces, cada C_i es de la forma $p_j \vee \neg p_j \vee \tilde{C}_i$, para algún $1 \leq j \leq n$ y alguna cláusula $\tilde{C}_i$.

Demostración. Escribamos $C_i = \varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_m$ con φ_k un literal o la negación de un literal i.e. $\varphi_k = p_{i_k}$ o $\varphi_k = \neg p_{i_k}$. Si algún C_i no es de la forma descrita en el enunciado, tenemos que no aparecen los átomos p_{i_k} y $\neg p_{i_k}$ simultaneamente. De este modo, consideremos la evaluación de verdad v que hace $v(p_{i_k}) = 0$ si $\varphi_k = p_{i_k}$ y $v(p_{i_k}) = 1$ si $\varphi_k = \neg p_{i_k}$. Con esa evaluación, tenemos que $v(C_i) = \text{máx}(v(\varphi_1), \ldots, v(\varphi_m)) = \text{máx}(0, \ldots, 0) = 0$. En consecuencia, $v(\psi) = \text{mín}(v(C_i)) = 0$ y, por tanto, ψ no puede ser una tautología semántica. □

Ejemplo 1.48. Pasemos a FNC la tautología $\psi = ((p \to q) \wedge p) \to q$. Para ello, tenemos

$$\begin{aligned}((p \to q) \wedge p) \to q &\equiv \neg((\neg p \vee q) \wedge p) \vee q \equiv (\neg(\neg p \vee q) \vee \neg p) \vee q \vee \equiv ((p \wedge \neg q) \vee \neg p) \vee q \\ &\equiv ((p \vee \neg p) \wedge (\neg q \vee \neg p)) \vee q \equiv (p \vee \neg p \vee q) \wedge (\neg q \vee \neg p \vee q) \\ &\equiv (p \vee \neg p \vee \underbrace{q}_{\tilde{C}_1}) \wedge (q \vee \neg q \vee \underbrace{\neg p}_{\tilde{C}_2}).\end{aligned}$$

Teorema 1.49 (Completitud)**.** *El sistema de deducción natural es completo.*

Demostración. Por la Proposición 1.45, basta probar que las tautologías semánticas son demostrables i.e. si $\models \psi$ entonces $\vdash \psi$. Para ello, usamos el Corolario 1.47 para obtener que $\vdash \psi \leftrightarrow \text{FND}(\psi)$. Como ψ es una tautología, por el Lema 1.47 se tiene que $\text{FND}(\psi) = \bigwedge_i p_{j_i} \vee \neg p_{j_i} \vee \tilde{C}_i \vdash p_j \vee \neg p_j \vee \tilde{C}_j$ con p_j cualquier literal de ψ que aparece en $\text{FND}(\psi)$. En particular, esto implica que

$$\vdash \neg\psi \to \neg(p_j \vee \neg p_j) \wedge \neg\tilde{C}_j \vdash \neg\psi \to \neg(p_j \vee \neg p_j) \vdash \neg\psi \to (p_j \wedge \neg p_j).$$

Usando $\neg^{in}$, se deduce que $\vdash \neg\neg\psi \vdash_{\neg^{out}} \psi$, luego ψ es derivable, como queríamos demostrar. □

Corolario 1.50. *Una argumento proposicional es cierto semánticamente (i.e. es verdad) si y solo si es cierto sintácticamente (i.e. tiene una demostración).*

1.5. Lógica computacional y aplicaciones a la IA. El óbjetivo de la lógica computacional es producir deducciones de forma automática. Esto permite razonar de forma autónoma sin mediación humana, lo que resulta enormemente útil en inteligencia artificial.

Una primera aproximación a la deducción automática puede obtenerse utilizando el Lema 1.47. Recordemos que, en virtud a este resultado, una fórmula ψ es una tautología si y solo si todas las cláusulas C de $\text{FNC}(\psi)$ son de la forma $C = p \vee \neg p \vee \tilde{C}$ para cierto literal p.

Data: Formula ψ
Result: Si ψ es una tautología o no
$\mathcal{C}$:= Conjunto de clausulas de $\text{FNC}(\psi)$;
for $C \in \mathcal{C}$ **do**
  if *C no es de la forma $p \vee \neg p \vee \tilde{C}$* **then**
    return Tautología;
  end
end
return No tautología;

Algorithm 1: Algoritmo naïve de detección de tautologías

En particular, usando este algoritmo, podemos obtener un sistema de deducción automática para teorías finitas. Supongamos que $T \subseteq$ **For** es una teoría finita y sea $\psi_T = \bigwedge_{\psi \in T} \psi$. Tenemos que $T \models \varphi$ si y solo si $\models \psi_T \to \varphi$, y esta última comprobación puede realizar usando el Algoritmo 1. No obstante, esta estrategia adolece de tres problemas fundamentales:

- Cada vez que se quiere comprobar $T \models \varphi$, es necesario computar la Forma Normal Conjuntiva de $\psi_T \to \varphi$. Esta FNC depende profundamente de φ y, por tanto, no es posible acelerar el proceso precomputando ninguna forma normal.
- La comprobación de que $\psi_T \to \varphi$ es una tautología requiere analizar **todas** las cláusulas de su FNC. Esto puede resultar computacionalmente infactible si T es muy grande.
- Esta estrategia no puede generalizarse a teorías infinitas, puesto que no es posible revisar todas sus cláusulas.

Para solventar estos problemas, la idea de Davis y Putnam fue reformular el Algoritmo 1 para comprobar insatisfacibilidad en lugar de tautoligicidad. Recuérdese que, para toda teoría $T \subseteq \mathbf{For}$ y toda fórmula $\psi \in \mathbf{For}$, se tiene que

$$T \models \psi \text{ si y solo si } T \cup \{\neg\psi\} \text{ es insatisfacible.}$$

En esta formulación, el problema de deducción automática se reduce a un problema de **comprobación de insatisficibilidad**.

Para comprobar esta insatisfacibilidad, Davis y Putnam proponen usar una regla de deducción derivada conocida como *regla de resolución*.

Proposición 1.51 (Regla de resolución)**.** *Para todo $\psi, \varphi \in \mathbf{For}$ y todo literal p, se tiene*

$$\psi \vee p, \varphi \vee \neg p \models \psi \vee \varphi.$$

Demostración. La comprobación con $\models$ es inmediata usando tablas de verdad. Si queremos comprobarlo sintácticamente con $\vdash$, usamos la siguiente derivación.

(1)	$\rightsquigarrow$		Inicio regla fantasiosa
(2)		$\neg\psi \wedge \neg\varphi$	Supuesto
(3)		$\neg\psi$	$\wedge_1^{out}(2)$
(4)		$\psi \vee p$	Premisa
(5)		p	C(4,3)
(6)		$\neg\varphi$	$\wedge_2^{out}(2)$
(7)		$\varphi \vee \neg p$	Premisa
(8)		$\neg p$	C(7,6)
(9)		$p \wedge \neg p$	$\wedge^{in}(5,8)$
(10)	$(\neg\psi \wedge \neg\varphi) \to (p \wedge \neg p)$		Fin regla fantasiosa $\to^{in}$ $(2,9)$
(11)	$\neg(\neg\psi \wedge \neg\varphi)$		$\neg^{in}(10)$
(12)	$\psi \vee \varphi$		dM

□

Dada una teoría $T \subseteq \mathbf{For}$, puede construirse su **forma clausular**, $\overline{T}$. Supongamos que escribimos cada fórmula $\psi \in T$ en su Forma Normal Disjuntiva

$$\mathrm{FND}(\psi) = \bigwedge_i C_i(\psi),$$

con $C_i(\psi)$ cláusulas. Entonces, $\overline{T}$ está dada por

$$\overline{T} = \left\{ C_i(\psi) \;\middle|\; \psi \in T, \mathrm{FND}(\psi) = \bigwedge_i C_i(\psi) \right\}.$$

Es decir, $\overline{T}$ es la teoría formada por todas las cláusulas de todas las FND de fórmulas de T. Obsérvese que, por construcción, T y $\overline{T}$ son equivalentes. Se dice que T **forma clausular** si $T = \overline{T}$.

Ejemplo 1.52. Pasar a forma clausular la teoría $T = \{p \to q, (q \to r) \wedge p, \neg r\}$.

Tenemos que las FND de las fórmulas de T son

$$\mathrm{FND}(p \to q) = \neg p \vee q, \quad \mathrm{FND}((q \to r) \wedge p) = (\neg q \vee r) \wedge p, \quad \mathrm{FND}(\neg r) = \neg r.$$

En consecuencia, la forma clausular es

$$\overline{T} = \{\neg p \vee q, \neg q \vee r, p, \neg r\}$$

Teorema 1.53 (Resolución es completa para refutación)**.** *Sea $T \subseteq$* **For** *una teoría en forma clausular. Entonces T es insatisfacible si y solo si, aplicando únicamente la regla de resolución, se puede obtener la cláusula vacía.*

Demostración. La clave de la demostración es probar que, si T es insatisfacible, entonces se puede aplicar la regla de resolución para reducir T. Como el número de veces que se puede reducir T es finito, finalmente se llega a la cláusula vacía. La prueba de este hecho es muy similar a la del Lema 1.47, considerando contradicciones en lugar de tautologías. □

Data: Teoría T
Result: Si T es satisfacible o no
$X :=$ Forma clausular de T;
while *not* $\emptyset \in X$ **do**
  $\mathcal{R}(X) :=$ Conjunto de todos los resolventes posibles de X;
  if $\mathcal{R}(X) = X$ **then**
    return T es satisfacible;
  end
  $X := \mathcal{R}(X)$;
end
return T es insatisfacible;

Algorithm 2: Algoritmo de Davis-Putnam

Observación 1.54. El algoritmo 2 no solo permite comprobar la insatisfacibilidad de T, sino también se puede devolver una asignación de verdad que satisfaga T en caso de que sea satisfacible. Obsérvese que, si $\mathcal{R}(X) = X$, entonces no se puede resolver en ninguna variable y, por tanto, para todo literal p, o aparece p o aparece $\neg p$ en las cláusulas de X (o $p \vee \neg p$ en una misma cláusula, que no altera el resultado). En cualquier caso, utilizando la técnica del Lema [REF], puede extraerse una asignación de verdad que satisfaga X, y por tanto T.

Ejemplo 1.55. Demostrar, mediante el algoritmo de Davis-Putnam que $p \to q, (q \to r) \wedge p \models r$.

En primer lugar, esta comprobación se reduce a que la teoría $T = \{p \to q, (q \to r) \wedge p, \neg r\}$ sea insatisfacible. La forma clausular de T fue computada en el Ejemplo 1.52 y es

$$\overline{T} = \{\neg p \vee q, \neg q \vee r, p, \neg r\}\,.$$

Aplicamos repetidas veces reducción, obteniendo

(1)	$\neg p \vee q$	Premisa
(2)	$\neg q \vee r$	Premisa
(3)	p	Premisa
(4)	$\neg r$	Premisa
(5)	$\neg p \vee r$	Reducción de (1) y (2) en q
(6)	r	Reducción de (3) y (5) en p
(7)	$\emptyset$	Reducción de (4) y (6) en r

Ejemplo 1.56. Dar una asignación de verdad que satisfaga a la teoría

$$T = \{p, q \vee \neg p \vee \neg r, r \vee s, \neg s\}\,.$$

T está en forma clausular, luego podemos aplicar el algoritmo de David-Putnam, obteniendo

(1)	p	Premisa
(2)	$q \vee \neg p \vee \neg r$	Premisa
(3)	$r \vee s$	Premisa
(4)	$\neg s$	Premisa
(5)	$q \vee \neg r$	Reducción de (1) y (2) en p
(6)	r	Reducción de (3) y (4) en s
(7)	q	Reducción de (5) y (6) en r

De este modo, $\mathcal{R}(T) = \{p, q, r, \neg s\} \cup \{\text{Otras fórmulas reducibles}\}$, por lo que la asignación de verdad $v(p) = v(q) = v(r) = 1$ y $v(s) = 0$ satisface T.

1.5.1. Forma de Horn y programación lógica. A pesar de que el algoritmo de Davis-Putnam permite comprobar automáticamente la satisfacibilidad o insatisfacibilidad de una fórmula, este algoritmo es equivalente a n-SAT y, por tanto, es un problema **NP**. Con el fin de obtener un algoritmo eficiente de satisfacibilidad, se pueden considerar un subconjunto de todas las fórmulas proposicionales, llamadas las **cláusulas de Horn**.

Definición 1.57. Una cláusula C se dice que es una **cláusula de Horn** si tiene, a lo sumo, un literal no negado. Si C tiene exactamente un literal no negado, se dice que C es una **cláusula de Horn estricta**. Si C tiene todos sus símbolos negados, se dice que C es una **cláusula objetivo**.

Ejemplo 1.58.
- $\neg p \vee q \vee \neg r$ es una cláusula de Horn estricta.
- $\neg p \vee \neg q \vee \neg r$ es una cláusula de Horn objetivo.
- $\neg p \vee q \vee r$ **no** es una cláusula de Horn.

Observación 1.59. A pesar de que escribiremos las cláusulas de Horn siempre como disjunción de átomos (para poder aplicar resolución), la forma más adecuada de pensar una cláusula de Horn es como una implicación. En efecto, la cláusula de Horn

$$C = \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q$$

es equivalente a la implicación

$$p_1 \wedge p_2 \wedge \ldots \wedge p_n \to q.$$

En programación lógica, es costumbre escribir esta última fórmula de derecha a izquierda como

$$q \leftarrow p_1 \wedge p_2 \wedge \ldots \wedge p_n$$

y entender que esa regla significa que, de que se cumpla $p_1, p_2, \ldots$ y p_n se sigue que se cumple q.

Observación 1.60. No todas las fórmulas son equivalentes a una colección de cláusulas de Horn (e.g. $\psi = p \vee q \vee \neg r$), por lo que la lógica de Horn es un subconjunto de la lógica proposicional.

Observación 1.61. Las cláusulas de Horn son cerradas para la resolución, esto es, la resolución de dos cláusulas de Horn es una cláusula de Horn.

Si trabajamos únicamente con cláusulas de Horn, existe una manera eficiente de aplicar resolución en el algoritmo de Davis-Putnam para comprobar la insatisfacibilidad de una teoría. Este el llamado **algoritmo SLD** (Selective Linear Definite), que tiene una complejidad lineal.

La idea del algoritmo es la siguiente. Supongamos que tenemos una teoría $T \subseteq$ **For** de cláusulas de Horn (que, en este contexto, se denomina la **base del conocimiento**) y queremos

probar que $T \models \psi$ para una cierta $\psi \in \mathbf{For}$ tal que $\neg\psi$ es una cláusula de Horn (generalmente, una cláusula objetivo). El algoritmo SLD escoge una cláusula $D_1 \in T$ y resuelve $C_1 = \neg\psi$ con D_1, obteniendo una cláusula de Horn C_2. Ahora escoge $D_2 \in T$ y resuelve C_2 con D_2, obteniendo C_3, y así sucesivamente. Si $C_n = \emptyset$, entonces $T \models \psi$. Por el contrario, si C_n no puede resolverse más, el algoritmo vuelve hacia atrás hasta encontrar una cláusula deducida C_m con $m < n$ que puede reducirse con un D'_m distinto del D_m con el que se redujo en primera instancia. Si no es posible encontrar tal C_m, el algoritmo termina indicando que $T \not\models \psi$. De este modo, es una búsqueda en árbol de la cláusula vacía, mediante el algoritmo de **primero en profundidad** (DFS).

Data: Base del conocimiento T y fórmula ψ con $\neg C$ una cláusula de Horn.
Result: **True** si $T \models \psi$ o **False** si no.
$C := \neg\psi$;
if $C = \emptyset$ **then**
  return True;
end
for $D \in T$ **do**
  if *D puede resolverse con C* ***and*** $\mathrm{SLD}(T, \mathrm{Resolver}(D, C)) =$ ***True*** **then**
    return True;
  end
end
return False;

Algorithm 3: Algoritmo SLD

2. Lógica de primer orden

Consideremos el siguiente enunciado (conocido como la conjetura de Goldbach).

"Todo número par mayor que 2 se escribe como la suma de dos número primos"

Definitivamente, la lógica proposicional es insuficiente para representar toda la complejidad de esta fórmula. Hay varios conceptos que no podemos capturar: "número par", "número primo", "suma", "todo número". Para ser capaces de formular estos enunciados, necesitaremos enriquecer enormemente la lógica con variables, constantes, funciones, relaciones y cuantificadores.

Por ejemplo, cuando introduzcamos adecuadamente todos estos términos, veremos que el enunciado anterior puede formularse, por ejemplo, como la siguiente fórmula de lógica de primer orden:

$$\forall x((\mathbf{Par}(x) \wedge x > 2) \to \exists y \exists z(\mathbf{Primo}(y) \wedge \mathbf{Primo}(z) \wedge (x = y + z))).$$

2.1. Formulación de la lógica de primer orden. La Lógica de Primer Orden, a veces denotada LPO, es un enriquecimiento de la lógica proposicional. Grosso modo, la lógica de primer orden generaliza a la lógica proposicional en dos sentidos:

- Los literales se descomponen para dar mucha más riqueza expresiva al lenguaje. De este modo, donde ahora teníamos un único literal, ahora tendremos: **átomos** que a su vez están formados en base a **términos**.
- Se permite la introducción de variables, que podremos cuantizar utilizando dos símbolos nuevos el **cuantificador universal**, $\forall$, y el **cuantificador existencial**, $\exists$.

El primer paso en la formulación de la lógica de primer orden es fijar un **lenguaje** L. Este consta de **símbolos** de tres tipos distintos:

- **Constantes**. Los denotaremos con $a, b, c, \ldots$ pero también como $0, 1, 27, \ldots$
- **Funciones**. Cada función tiene asociada una **aridad**, que es un entero postivo que representa el número de argumentos que la función admite. Las funciones las llamaremos $f, g, h, \ldots$ pero también $+, \cdot, ^{-1}, \ldots$
- **Relaciones**. Nuevamente, una relación tiene asociada una **aridad**, que representa el número de elementos que relaciona. Las relaciones las denotaremos $R, G, H, \ldots$, pero también $>, <, \geq, \ldots$ Siempre tendremos una relación de aridad 2 (binaria), $=$, que representa la igualdad.

Además del lenguaje, tenemos a nuestra disposición una cantidad infinita de **variables**, que denotaremos $x, y, z, t, \ldots$ Las variables deben entenderse en el mismo sentido que en programación: son huecos en los que deberemos insertar un valor concreto cuando vayamos a evaluar la fórmula.

Definición 2.1. A partir del lenguaje L y de las variables, podemos formar los **términos**. Estos se construyen recursivamente como:

1. Una variable es un término.
2. Una constante es un término.
3. Si f es una función de aridad $n \geq 1$ y $t_1, \ldots, t_n$ son términos, entonces $f(t_1, \ldots, t_n)$ es un término.

Ejemplo 2.2. Supongamos que nuestro lenguaje L tiene por constantes los números naturales y como funciones: f y $\cdot^{-1}$ de aridad 1, $+$ y $\cdot$ de aridad 2 y g de aridad 3. Entonces los siguientes son términos:

1. 2.
2. $x + 2$.
3. $f(x + 2) \cdot y^{-1}$.
4. $g(1, f(x + 2) \cdot y^{-1}, t + u) + 5^{-1}$.

Ahora, a partir de los términos, podemos construir los átomos, también llamadas proposiciones. La idea clave es que los átomos son la unidad mínima que toma un valor de verdad. En este sentido, los átomos juegan el papel análogo a los literales en la lógica proposicionas.

Definición 2.3. Sea R es una relación de aridad n y sean $t_1, \ldots, t_n$ términos. Entonces, $R(t_1, \ldots, t_n)$ es un **átomo**.

Ejemplo 2.4. Con el lenguaje anterior, añadimos los símbolos de relación: **pos** de aridad 1 y $>$ de aridad 2, además del sempiterno $=$. Los siguientes son átomos:

1. $2 > 0$.
2. $\mathbf{pos}(x + 2)$.
3. $f(x + 2) \cdot y^{-1} > x + f(1)$.
4. $\mathbf{pos}(f(x + 2) \cdot y^{-1})$.
5. $f(x + 2) \cdot y^{-1} = (z \cdot t)^{-1} + 2$.

Observación 2.5. En muchas ocasiones, querremos enfatizar las variables que aparecen en un átomo a. Para ello, si $x_1, \ldots, x_n$ son las variables que aparecen en a, entonces escribiremos $a(x_1, \ldots, x_n)$. Por ejemplo, para el átomo $a = f(x + 2) \cdot y^{-1} = (z \cdot t)^{-1} + 2$ escribiremos $a(x, y, z, t)$.

En este punto, las fórmulas se construyen de igual forma que en la lógica proposicional, utilizando los mismos operadores: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$. No obstante, a estos símbolos añadiremos dos

cuantificadores: $\forall$ y $\exists$. Al contrario que las conectivas proposicionales, los cuantificadores tienen dos partes diferenciadas: la variable que cuantifican y la fórmula que cuantifican.

Definición 2.6. El conjunto de **fórmulas** de la lógica de primer orden, escrito $\mathbf{For}_{\mathrm{LPO}}$ o simplemente **For** cuando no se confunde con la lógica proposicional, se construye recursivamente de la siguiente forma:

- Si a es un átomos, entonces $a \in \mathbf{For}_{\mathrm{LPO}}$.
- Si $\psi, \varphi \in \mathbf{For}_{\mathrm{LPO}}$, entonces:
  - $\psi \wedge \varphi \in \mathbf{For}_{\mathrm{LPO}}$.
  - $\psi \vee \varphi \in \mathbf{For}_{\mathrm{LPO}}$.
  - $\psi \to \varphi \in \mathbf{For}_{\mathrm{LPO}}$.
  - $\psi \leftrightarrow \varphi \in \mathbf{For}_{\mathrm{LPO}}$.
  - $\neg\psi \in \mathbf{For}_{\mathrm{LPO}}$.
- Si $\psi \in \mathbf{For}_{\mathrm{LPO}}$ y x es una variable, entonces:
  - $\forall x\psi \in \mathbf{For}_{\mathrm{LPO}}$.
  - $\exists x\psi \in \mathbf{For}_{\mathrm{LPO}}$.

Observación 2.7. Como con la lógica proposicional, utilizaremos paréntesis para desambiguar las fórmulas cuando sea necesario.

Ejemplo 2.8. Con el lenguaje de los ejemplos anteriores, las siguientes son fórmulas:

1. $((2 > 0) \wedge \neg\mathbf{pos}(x + 2)) \to (x + f(1))$.
2. $\mathbf{pos}(x + 2) \to (y > 1)$.
3. $(\exists x\mathbf{pos}(x + 2)) \to (y > 1)$.
4. $\forall x\,(\mathbf{pos}(x + 2) \to (y > 1))$.
5. $\forall x\,(\mathbf{pos}(x + 2) \to (x + y > 1))$.
6. $\forall y\,(\mathbf{pos}(x + 2) \to (y > 1))$.
7. $\forall y \exists x\,(\mathbf{pos}(x + 2) \to (y > 1))$.
8. $\exists y\,(\mathbf{pos}(x + 2) \to (y > 1))$.
9. $\forall y\,((\exists x\mathbf{pos}(x + 2)) \to (y > 1))$.
10. $(\forall x(\mathbf{pos}(x) > 2)) \vee (\exists y(f(y) = 3))$.

Dada una fórmula ψ, una variable x se dice **ligada** si aparece bajo el ámbito de algún cuantificador de la forma $\forall x$ o $\exists x$. En caso contrario, se dice que x es una variable **libre**. Si, dada una fórmula ψ, queremos enfatizar que sus variables libres son $x_1, \ldots, x_n$ escribiremos $\psi(x_1, \ldots, x_n)$.

Definición 2.9. Una fórmula se dice **cerrada** si no contiene variables libres. En caso contrario, la fórmula se dice **abierta**. El conjunto de fórmulas cerradas se denotará por **CFor**.

2.1.1. Ejemplos de formalización en lógica de primer orden.

Ejemplo 2.10. Consideremos el lenguaje que tiene:

- Constantes: $\mathrm{luis}, \mathrm{laura}$ y $\mathrm{ps5}$.
- Funciones: No hay.
- Relaciones: Comprar.

Entonces la formalización de los siguientes enunciados es:

1. "Luis compra una PS5"

$$\mathrm{Comprar}(\mathrm{luis}, \mathrm{ps5}).$$

2. "Luis compra algo"
$$\exists x \text{Comprar}(\text{luis}, x).$$
3. "Laura compra todo lo que compra Luis".
$$\forall x(\text{Comprar}(\text{luis}, x) \to \text{Comprar}(\text{laura}, x)).$$
4. "Si Luis compra todo, entonces Laura también lo hace".
$$(\forall x \text{Comprar}(\text{luis}, x)) \to (\forall x \text{Comprar}(\text{laura}, x)).$$
5. "Si Luis compra todo, entonces Laura también compra algo".
$$(\forall x \text{Comprar}(\text{luis}, x)) \to (\exists x \text{Comprar}(\text{laura}, x)).$$
6. "Todo el mundo compra algo".
$$\forall x \exists y \text{Comprar}(x, y).$$
7. "Alguien compra todo".
$$\exists x \forall y \text{Comprar}(x, y).$$

Observación 2.11. El nombre concreto de las variables es irrelevante y se pueden renombrar. Se dice que son variables mudas, no representan nada concreto, simplemente un hueco donde insertar un valor posteriormente. De este modo, si en alguna fórmula queda ambiguo el papel de una variable (por ejemplo, porque se solapa el ámbito de dos cuantificadores), podemos simplemente renombrar las variables para desambiguar.

2.2. Modelos y consecuencia semántica. En esta sección, trataremos la verdad semántica en la lógica de primer orden. Al igual que en el caso de la lógica proposicional, la veracidad o falsedad de una fórmula no es, en general, absoluta, sino que depende de una determinada "asignación de verdad". No obstante, mientras que las asignaciones de verdad en lógica proposicional eran muy sencillas (básicamente, consistían en asignar un valor de verdad a los literales), en el caso de la lógica de primer orden necesitamos un concepto mucho más rico: los modelos.

Definición 2.12. Sea L un lenguaje de lógica de primer orden. Un **modelo** para L, también llamado una **interpretación**, es una tupla $\mathcal{A} = (A, \iota)$, donde A es un conjunto, llamado el **dominio**, y ι es una asignación tal que:

- A toda constante c de L, asigna un elemento $\iota(c) \in A$.
- A toda función f de L de aridad n, asigna una función $\iota(f) : A^n \to A$.
- A toda relación R de L de aridad n, asigna un subconjunto $\iota(R) \subseteq A^n$.

Observación 2.13. Es costumbre sobreentender el dominio A y la asignación ι, y hacer referencia únicamente al modelo $\mathcal{A}$. En ese caso, la interpretación de una constante c, una función f y una relación R se denotan $c^{\mathcal{A}}$, $f^{\mathcal{A}}$ y $R^{\mathcal{A}}$.

En principio, un modelo únicamente asigna valores a los símbolos del lenguaje. No obstante, operando recursivamente, podemos dar asignar un elemento del dominio a cualquier término sin variables libres, como muestra la siguiente definición.

Definición 2.14. Sea L un lenguaje de la lógica de primer orden y $\mathcal{A} = (A, \iota)$ un modelo para L. Entonces, dado un término t sin variables, podemos asignarle un valor $t^{\mathcal{A}} \in A$ del siguiente modo:

- Si $t = c$ es una constante, $t^{\mathcal{A}} = c^{\mathcal{A}}$.

- Si $t = f(t_1, \ldots, t_n)$, donde f es una función de aridad n y $t_1, \ldots, t_n$ son términos sin variables, obtenemos $t^{\mathcal{A}}$ como el resultado de aplicar la función $f^{\mathcal{A}}$ a cada uno de los elementos $t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}$, esto es $t^{\mathcal{A}} = f^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}})$.

Como veremos a continuación, con el fin de formular la definición de asignación de verdad, momentáneamente debemos considerar términos ligeramente más generales. Dado un modelo $\mathcal{A} = (A, \iota)$, diremos que t es un **término parametrizado por** $\mathcal{A}$ si t no contiene variables pero, en su lugar, puede contener apariciones de elementos de A. Dicho de otro modo, son términos que se construyen como en la Definición 2.1, pero sustituyendo la regla 1 por "Si $a \in A$ es un elemento del dominio, entonces a es un término". Obsérvese que, si t es un término parametrizado por $\mathcal{A}$, podemos asignarle un valor $t^{\mathcal{A}} \in A$ simplemente añadiendo la regla obvia que dice que, si $t = a$ para algún $a \in A$, entonces $t^{\mathcal{A}} = a$.

De igual forma, si ψ es una fórmula construida a partir de términos en $\mathcal{A}$, diremos que ψ es una **fórmula parametrizada por** $\mathcal{A}$.

Definición 2.15. Dado un modelo $\mathcal{A} = (A, \iota)$, $a \in A$ y $\psi = \psi(x)$ una fórmula con la variable x libre, denotaremos por $\psi(a)$ la fórmula parametrizada por $\mathcal{A}$ que resulta de substituir todas las apariciones de x en ψ por a.

Definición 2.16. Sea L un lenguaje y $\mathcal{A}$ un modelo para L. Entonces, tenemos una función

$$v_{\mathcal{A}} : \mathbf{CFor} \to \{0, 1\}$$

definida de la siguiente forma:

- Si $\psi = R(t_1, \ldots, t_n)$, con t_i términos parametrizados por $\mathcal{A}$, entonces $v_{\mathcal{A}}(\psi) = 1$ si tenemos que $(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}) \in R^{\mathcal{A}}$ y $v_{\mathcal{A}}(\psi) = 0$ si no.
- Para fórmulas de la forma $\psi \wedge \varphi$ con $\psi, \varphi \in \mathbf{CFor}$

$$v_{\mathcal{A}}(\psi \wedge \varphi) = \text{mín}(v_{\mathcal{A}}(\psi), v_{\mathcal{A}}(\varphi)) = \begin{cases} 1 & \text{si } v_{\mathcal{A}}(\psi) = 1 \text{ y } v_{\mathcal{A}}(\varphi) = 1 \\ 0 & \text{si } v_{\mathcal{A}}(\psi) = 0 \text{ o } v_{\mathcal{A}}(\varphi) = 0 \end{cases},$$

- Para fórmulas de la forma $\psi \vee \varphi$ con $\psi, \varphi \in \mathbf{CFor}$

$$v_{\mathcal{A}}(\psi \vee \varphi) = \text{máx}(v_{\mathcal{A}}(\psi), v_{\mathcal{A}}(\varphi)) = \begin{cases} 1 & \text{si } v_{\mathcal{A}}(\psi) = 1 \text{ o } v_{\mathcal{A}}(\varphi) = 1 \\ 0 & \text{si } v_{\mathcal{A}}(\psi) = 0 \text{ y } v_{\mathcal{A}}(\varphi) = 0 \end{cases},$$

- Para fórmulas de la forma $\psi \to \varphi$ con $\psi, \varphi \in \mathbf{CFor}$

$$v_{\mathcal{A}}(\psi \to \varphi) = \begin{cases} 1 & \text{si } v_{\mathcal{A}}(\psi) = 0 \text{ o } (v_{\mathcal{A}}(\psi) = 1 \text{ y } v_{\mathcal{A}}(\varphi) = 1) \\ 0 & \text{si } v_{\mathcal{A}}(\psi) = 1 \text{ y } v_{\mathcal{A}}(\varphi) = 0 \end{cases},$$

- Para fórmulas de la forma $\psi \leftrightarrow \varphi$ con $\psi, \varphi \in \mathbf{CFor}$

$$v_{\mathcal{A}}(\psi \leftrightarrow \varphi) = \begin{cases} 1 & \text{si } v_{\mathcal{A}}(\psi) = v_{\mathcal{A}}(\varphi) \\ 0 & \text{si si } v_{\mathcal{A}}(\psi) \neq v_{\mathcal{A}}(\varphi) \end{cases},$$

- Para fórmulas de la forma $\neg\psi$ con $\psi \in \mathbf{CFor}$,

$$v_{\mathcal{A}}(\neg\psi) = 1 - v_{\mathcal{A}}(\psi).$$

- Para fórmulas de la forma $\forall x\psi$ con $\psi = \psi(x)$, entonces $v_{\mathcal{A}}(\forall x\psi) = 1$ si $v_{\mathcal{A}}(\psi(a)) = 1$ para todo $a \in A$, en caso contrario $v_{\mathcal{A}}(\forall x\psi) = 0$.
- Para fórmulas de la forma $\exists x\psi$ con $\psi = \psi(x)$, entonces $v_{\mathcal{A}}(\exists x\psi) = 1$ si $v_{\mathcal{A}}(\psi(a)) = 1$ para algún $a \in A$, en caso contrario $v_{\mathcal{A}}(\exists x\psi) = 0$.

Si $v_{\mathcal{A}}(\psi) = 1$, diremos que $\mathcal{A}$ **satisface** ψ, o que ψ es **cierta** en $\mathcal{A}$, y lo denotaremos por $\mathcal{A} \models \psi$.

Al igual que hicimos en lógica proposicional, una vez hemos definido el concepto de verdad semántica, podemos definir qué significa que una fórmula sea consecuencia semántica. Análogamente a la versión proposicional, una **teoría** es un subconjunto $T \subseteq \mathbf{CFor}$ de fórmulas cerradas.

Definición 2.17. Sea T una teoría y $\psi \in \mathbf{CFor}$. Diremos que T **implica semánticamente** ψ, denotado $T \models \psi$, si para todo modelo $\mathcal{A}$ con $\mathcal{A} \models \varphi$ para todo $\varphi \in T$, se tiene que $\mathcal{A} \models \psi$.

Observación 2.18. Como siempre, es posible que $T = \emptyset$ sea la teoría vacía. En ese caso, denotamos $\models \psi$ y decimos que ψ es una **tautología**.

Observación 2.19. Al igual que en lógica proposicional, si $T = \{\varphi\}$ es una teoría con una única fórmula, la implicación semántica se puede reformular como que $\models \varphi \to \psi$.

De igual forma, una vez que tenemos el concepto de verdad, de forma inmediata obtenemos el concepto de equivalencia semántica.

Definición 2.20. Sean $\psi, \varphi \in \mathbf{CFor}$ dos fórmulas cerradas. Diremos que ψ y φ son **equivalentes semánticamente**, y lo denotaremos $\psi \equiv \varphi$ si para todo modelo $\mathcal{A}$ se tiene que $\mathcal{A} \models \psi$ si y solo si $\mathcal{A} \models \varphi$.

Observación 2.21. Al igual que en lógica proposicional, la equivalencia semántica se puede reformular como que $\psi \models \varphi$ y $\varphi \models \psi$; o más sucintamente que $\models \psi \leftrightarrow \varphi$.

Proposición 2.22. *Son fórmulas equivalentes las siguientes:*

- *Si $\psi(x_1, \ldots, x_n), \varphi(x_1, \ldots, x_n)$ son dos fórmulas sin cuantificadores que, entendiendo sus átomos como literales, son equivalentes como fórmulas de lógica proposicional, entonces*

$$Q_1 x_1\, Q_2 x_2 \ \ldots Q_n x_n\, \psi(x_1, \ldots, x_n) \equiv C_1 x_1\, C_2 x_2 \ \ldots C_n x_n \varphi(x_1, \ldots, x_n)$$

  para cualesquiera cuantificadores $Q_i = \forall, \exists$.
- *Si $\psi(x)$ es una fórmula con variable libre x e y es una variable que no aparece en ψ, entonces*

$$\forall x\, \psi(x) \equiv \forall y\, \psi(y), \quad \exists x\, \psi(x) \equiv \exists y\, \psi(y).$$

- *Para toda fórmula $\psi = \psi(x) \in \mathbf{For}$,*

$$\neg \forall x\, \psi(x) \equiv \exists x\, \neg \psi(x), \quad \neg \exists x\, \psi(x) \equiv \forall x\, \neg \psi(x).$$

- *Para cualesquiera fórmulas $\psi = \psi(x) \in \mathbf{For}$ y $\varphi \in \mathbf{For}$, donde φ no tiene x libre, tenemos*

$$\begin{array}{ll} \forall x\, \psi(x) \wedge \varphi \equiv \forall x\, (\psi(x) \wedge \varphi) & \exists x\, \psi(x) \wedge \varphi \equiv \exists x\, (\psi(x) \wedge \varphi) \\ \forall x\, \psi(x) \vee \varphi \equiv \forall x\, (\psi(x) \vee \varphi) & \exists x\, \psi(x) \vee \varphi \equiv \exists x\, (\psi(x) \vee \varphi) \end{array}$$

- *Para cualesquiera fórmulas $\psi = \psi(x) \in \mathbf{For}$ y $\varphi = \varphi(y) \in \mathbf{For}$, tenemos*

$$\forall x\, \psi(x) \wedge \forall y\, \varphi(y) \equiv \forall z\, \left(\psi(z) \wedge \varphi(z)\right), \quad \exists x\, \psi(x) \vee \exists y\, \varphi(y) \equiv \exists z\, \left(\psi(z) \vee \varphi(z)\right).$$

Corolario 2.23. *A partir de estas equivalencias, podemos extraer otras equivalencias interesantes.*

- *Para cualesquiera fórmulas $\psi = \psi(x) \in \mathbf{For}$ y $\varphi \in \mathbf{For}$, donde φ no tiene x libre, tenemos*

$$\begin{array}{ll} \forall x\, \psi(x) \to \varphi \equiv \exists x\, (\psi(x) \to \varphi) & \exists x\, \psi(x) \to \varphi \equiv \forall x\, (\psi(x) \to \varphi) \\ \varphi \to \forall x\, \psi(x) \equiv \exists x\, (\varphi \to \psi(x)) & \varphi \to \exists x\, \psi(x) \equiv \forall x\, (\varphi \to \psi(x)) \end{array}$$

- *Para cualesquiera fórmulas $\psi = \psi(x) \in \mathbf{For}$ y $\varphi = \varphi(y) \in \mathbf{For}$, tenemos*

$$\begin{array}{ll} \forall x\, \psi(x) \vee \forall y\, \varphi(y) \equiv \forall x\, \forall y\ (\psi(x) \vee \varphi(y)) & \exists x\, \psi(x) \wedge \exists y\, \varphi(y) \equiv \exists x\, \exists y\ (\psi(x) \wedge \varphi(y)) \\ \forall x\, \psi(x) \vee \exists y\, \varphi(y) \equiv \forall x\, \exists y\ (\psi(x) \vee \varphi(y)) & \forall x\, \psi(x) \wedge \exists y\, \varphi(y) \equiv \forall x\, \exists y\ (\psi(x) \wedge \varphi(y)) \end{array}$$

2.2.1. Forma normal prénex. En lógica de primer orden, también es posible buscar formas normales de fórmulas. Esto es, dada una fórmula $\psi \in \mathbf{For}$, nuestro objetivo es buscar una fórmula equivalente $\psi' \equiv \psi$ con una cierta forma 'sencilla', 'canónica', que nos simplifique la deducción automática. En el caso de la lógica proposicional, nuestras formas normales fueron la Forma Normal Disyuntiva y, sobre todo, la Forma Normal Conjuntiva.

En el caso de la lógica de primer orden, existen extensiones naturales de estas formas normales. Naturalmente, la clave es tratar los nuevos símbolos lógicos que las fórmulas de primer orden añaden, esto es, los cuantificadores. En primer lugar, nos centraremos en la llamada forma prénex.

Definición 2.24. Una fórmula cerrada $\psi \in \mathbf{CFor}$ se dice que está en **forma prénex** si es de la forma

$$\psi = Q_1 x_1 \, Q_2 x_2 \, \dots Q_n x_n \, \varphi(x_1, \dots, x_n),$$

donde $Q_i = \forall, \exists$ son cuantificadores y φ es una fórmula sin cuantificadores con variables libre $x_1, x_2, \dots, x_n$.

Proposición 2.25. *Dada una fórmula $\psi \in \mathbf{CFor}$, existe una fórmula* $\mathrm{Prenex}(\psi) \in \mathbf{CFor}$ *en forma prénex con $\psi \equiv \mathrm{Prenex}(\psi)$. Esta fórmula se llama la* ***forma normal prénex de*** *ψ.*

Demostración. Utilizar las equivalencias de la Proposición 2.22 y el Corolario 2.23 para empujar los cuantificadores hacia afuera, hasta que la fórmula se encuentre en forma prénex. □

Ejemplo 2.26. Encontremos la forma normal prénex de la fórmula

$$\forall x \left((\exists y R(x,y) \wedge \forall y \neg S(x,y)) \to \neg \forall y R(x,y) \right).$$

Para ello, utilizamos la siguiente cadena de equivalencias

$$\begin{aligned}
&\forall x \left((\exists y R(x,y) \wedge \forall y \neg S(x,y)) \to \neg \forall y R(x,y) \right) \overset{\text{Sig. impl.}}{\equiv} \forall x \left(\neg \left(\exists y R(x,y) \wedge \forall y \neg S(x,y) \right) \vee \neg \forall y R(x,y) \right) \\
&\overset{\text{dM}\wedge}{\equiv} \forall x \left(\neg \exists y R(x,y) \vee \neg \forall y \neg S(x,y) \vee \neg \forall y R(x,y) \right) \overset{\text{Intr. neg.}}{\equiv} \forall x \left(\forall y \neg R(x,y) \vee \exists y S(x,y) \vee \exists y \neg R(x,y) \right) \\
&\overset{\text{Sac.}\exists}{\equiv} \forall x \left(\forall y \neg R(x,y) \vee \exists y \left(S(x,y) \vee \neg R(x,y) \right) \right) \overset{\text{Ren.}\exists}{\equiv} \forall x \left(\forall y \neg R(x,y) \vee \exists z \left(S(x,z) \vee \neg R(x,z) \right) \right) \\
&\overset{\text{Sac.}\forall}{\equiv} \forall x \forall y \ \left(\neg R(x,y) \vee \exists z (S(x,z) \vee \neg R(x,z)) \right) \overset{\text{Sac.}\exists}{\equiv} \forall x \forall y \exists z \ \left(\neg R(x,y) \vee S(x,z) \vee \neg R(x,z) \right).
\end{aligned}$$

2.2.2. Forma normal de Skolem. La forma normal de Skolem es un paso más en la simplificación de una fórmula de lógica de primer orden. Como veremos, en este caso, la forma de Skolem **no es equivalente** a la fórmula original. Sin embargo, es equivalente para satisfacibilidad, esto es, una fórmula es satisfacible si y solo si su forma de Skolem lo es.

Definición 2.27. Una fórmula $\psi \in \mathbf{CFor}$ se dice que está en **forma de Skolem** si:

1. ψ está en forma prénex.
2. La parte proposicional (sin cuantificadores) de ψ está en FNC.
3. ψ solo tiene cuantificadores universales.

Dicho de otro modo, una fórmula en forma de Skolem es de la forma

$$\psi = \forall x_1 \, \forall x_2 \, \dots \forall x_n \, \varphi(x_1, \dots, x_n),$$

con φ en Forma Normal Conjuntiva.

Dada una fórmula ψ, es posible encontrar una fórmula $\psi' \equiv \psi$ con las propiedades 1. y 2. En efecto, basta considerar $\psi' = \mathrm{Prenex}(\psi)$ y, luego, trabajar con la parte proposicional para pasarla

a Forma Normal Conjuntiva, como se explicó en la Sección 1.2.3. En virtud de la Proposición 2.22, la fórmula ψ' satisfaciendo 1. y 2. es equivalente a ψ.

No obstante, para alcanzar la propiedad 3. es necesario aplicar un nuevo proceso a Prenex(ψ) conocido como **skolemización**. Este consiste en eliminar una cuantificación universal de una variable y. Para ello, se añade un nuevo símbolo de función f_y de aridad n, donde n es el número de variables cuantificadas delante de y en Prenex(ψ), digamos $x_1, \ldots, x_n$. Entonces, se substituye cada aparición de y en Prenex(ψ) por el término $f_y(x_1, \ldots, x_n)$. De forma más precisa, el proceso de skolemización en la variable y transforma

$$\begin{aligned}&\forall x_1\, \forall x_2\, \forall x_n\, \exists \boldsymbol{y}\, Q_1 z_1\, Q_2\, z_2 \ldots Q_m\, z_m\, \varphi(x_i, \boldsymbol{y}, z_i)\\ \rightsquigarrow\ &\forall x_1\, \forall x_2\, \forall x_n\, Q_1 z_1\, Q_2\, z_2 \ldots Q_m\, z_m\, \varphi(x_i, f_y(x_1, \ldots, x_n), z_i),\end{aligned}$$

donde $Q_i = \forall, \exists$ son cualesquiera cuantificadores.

Aplicando sucesivamente este proceso a todas las variables cuantificadas existencialmente de fuera a adentro, obtenemos una nueva fórmula, denotada Skolem(ψ), que se denomina la **forma normal de Skolem de** ψ.

Ejemplo 2.28. Pasar a forma normal de Skolem la fórmula

$$\forall x \exists y P(x, y).$$

Observemos que esta fórmula ya satisface 1. y 2. Para cumplir 3. skolemizamos la variable y, substituyéndola por $f_y(x)$, para obtener

$$\forall x P(x, f_y(x)).$$

Ejemplo 2.29. Pasar a forma normal de Skolem la fórmula

$$\forall x \exists y \forall z \exists t P(x, y, z, t).$$

Observemos que esta fórmula ya satisface 1. y 2. Para cumplir 3. skolemizamos la variable y, substituyéndola por $f_y(x)$, para obtener

$$\forall x \forall z \exists t P(x, f_y(x), z, t).$$

Aún es necesario skolemizar en t, que sigue cuantificada existencialmente. Subtituyéndola por $f_t(x, z)$ obtenemos la forma de Skolem

$$\forall x \forall z P(x, f_y(x), z, f_t(x, z)).$$

Ejemplo 2.30. Pasar a forma normal de Skolem la fórmula

$$\forall x\, (P(x) \to \exists y\, (Q(y) \wedge R(x, y)))$$

Para ello, en primer lugar la pasamos a forma prénex para satisfacer 1.

$$\begin{aligned}\forall x\, (P(x) \to \exists y\, (Q(y) \wedge R(x, y))) &\overset{\text{Sig. impl.}}{\equiv} \forall x\, (\neg P(x) \vee \exists y\, (Q(y) \wedge R(x, y)))\\ &\overset{\text{Sac.}\exists}{\equiv} \forall x \exists y\, (\neg P(x) \vee (Q(y) \wedge R(x, y)))\end{aligned}$$

Después, pasamos la parte proposicional de la fórmula a FNC para satisfacer 2.

$$\forall x \exists y\, (\neg P(x) \vee (Q(y) \wedge R(x, y))) \overset{\text{Dist.}\exists}{\equiv} \forall x \exists y\, ((\neg P(x) \vee Q(y)) \wedge (\neg P(x) \vee R(x, y)))$$

Finalmente, para cumplir 3. skolemizamos la variable y, substituyéndola por $f_y(x)$, para obtener

$$\forall x \exists y\, ((\neg P(x) \vee Q(y)) \wedge (\neg P(x) \vee R(x, y))) \rightsquigarrow \forall x\, ((\neg P(x) \vee Q(f_y(x))) \wedge (\neg P(x) \vee R(x, f_y(x))))$$

Observación 2.31. La fórmula original y su forma de Skolem no tienen por qué ser equivalentes. Por ejemplo, en el ejemplo 2.28, la fórmula original y su forma de Skolem no son equivalentes. No obstante, lo que sí se tiene es que son **equivalentes para satisfacibilidad**, es decir, existe un modelo $\mathcal{A}$ tal que $\mathcal{A} \models \psi$ si y solo si existe un modelo $\mathcal{B}$ tal que $\mathcal{B} \models \text{Skolem}(\psi)$ (¡pero $\mathcal{A}$ y $\mathcal{B}$ pueden ser distintos!).

2.3. Sistema de deducción natural en Lógica de Primer Orden. En esta Sección, extenderemos el sistema de deducción natural de Lógica Proposicional de la Sección [REF] para incluir las fórmulas de la Lógica de Primer Orden. Mediante este sistema, podemos realizar razonamientos **sintácticos**, que pueden ser implementados computacionalmente para llevar a cabo deducción automática.

El sistema de deducción natural en lógica de primer orden, al igual que su análogo proposicional, **no tiene axiomas**. Además, todas las reglas de deducción proposicionales siguen siendo válidas en lógica de primer orden.

De este modo, para cualesquiera fórmulas $\psi, \varphi \in \mathbf{For}_{\text{LPO}}$, se tienen las siguientes reglas de deducción.

$(\wedge^{in})$ $\psi, \varphi \rightarrowtail \psi \wedge \varphi$.
$(\wedge_1^{out})$ $\psi \wedge \varphi \rightarrowtail \psi$.
$(\wedge_2^{out})$ $\psi \wedge \varphi \rightarrowtail \varphi$.
$(\vee_1^{in})$ $\psi \rightarrowtail \psi \vee \varphi$.
$(\vee_2^{in})$ $\varphi \rightarrowtail \psi \vee \varphi$.
$(\vee^{out})$ $\psi \to \sigma, \varphi \to \sigma, \psi \vee \varphi \rightarrowtail \sigma$.
$(\neg^{in})$ $\psi \to (\varphi \wedge \neg\varphi) \rightarrowtail \neg\psi$ (Demostración por contradicción).
$(\neg^{out})$ $\neg\neg\psi \rightarrowtail \psi$ (Doble negación).
$(\to^{in})$ Para todo $T \subseteq \mathbf{For}$, si $T \cup \{\psi\} \vdash \varphi$ entonces $T \rightarrowtail \psi \to \varphi$ (Regla fantasiosa).
$(\to^{out})$ $\psi \to \varphi, \psi \rightarrowtail \varphi$ (Modus ponens).
$(\leftrightarrow^{out})$ $\psi \leftrightarrow \varphi \rightarrowtail (\psi \to \varphi) \wedge (\varphi \to \psi)$.
$(\leftrightarrow^{in})$ $(\psi \to \varphi) \wedge (\varphi \to \psi) \rightarrowtail \psi \leftrightarrow \varphi$.

La clave de la extensión del sistema de deducción natural es que, a parte de estas reglas proposicionales, incluímos cuatro reglas nuevas para tratar el **cuantificador existencial cuantificador universal**, una de introducción y otra de extracción para cada uno.

$(\exists^{in})$ Si t es un término y x es una variable que no aparece en ψ, entonces $\psi(t) \rightarrowtail \exists x\, \psi(x)$.
$(\exists^{out})$ Sea a_0 una nueva constante que no aparece en L, llamada **constante temporal**. Para todo $T \subseteq \mathbf{For}$, si $T \cup \{\psi(a_0)\} \vdash \varphi$, donde φ es una fórmula que no contiene a_0, entonces $T \cup \{\exists x\, \psi(x)\} \rightarrowtail \varphi$.
$(\forall^{out})$ Si t es un término con variables no ambiguas en ψ y x es una variable, entonces $\forall x\, \psi(x) \rightarrowtail \psi(t)$.
$(\forall^{in})$ Si x es una variable libre en ψ y ψ **no contiene variables temporales**, entonces $\psi(x) \rightarrowtail \forall x\, \psi(x)$.

Observación 2.32.
- Para poder aplicar estas reglas, siempre es necesario que la variable x no sea ambigua en la fórmula ψ.
- La regla $(\exists^{out})$ debe entenderse como que la fórmula $\exists x\, \psi(x)$ puede sustituirse por $\psi(a_0)$ para un nombre a_0 que denote el elemento que cumple ψ (sea quien sea).
- La regla $(\forall^{in})$ debe interpretar se como si la afirmación $\psi(x)$ que es cierta para un x arbitrario, entonces $\forall x\, \psi(x)$ es cierto.

Ejemplo 2.33. Probar que

$$P(a) \to Q(a), \forall x\, P(x) \vdash Q(a)$$

(1)	$\forall x\, P(x)$	Premisa
(2)	$P(a)$	$\forall^{out}(2)$
(3)	$P(a) \to Q(a)$	Premisa
(4)	$Q(a)$	$\to^{out}(2,3)$

Ejemplo 2.34. Probar que

$$\forall x\, (P(x) \wedge Q(x)) \vdash \exists x\, P(x)$$

(1)	$\forall x\, (P(x) \wedge Q(x))$	Premisa
(2)	$P(a) \wedge Q(a)$	$\forall^{out}(2)$
(3)	$P(a)$	$\wedge_1^{out}(2)$
(4)	$\exists x\, P(x)$	$\exists^{in}(3)$

Ejemplo 2.35. Para aplicar la regla ($\exists^{out}$), utilizaremos la misma notación de indentación que en la regla fantasiosa. Por ejemplo, supongamos que queremos demostrar que

$$\exists x\, (P(x) \wedge Q(x)) \vdash \exists x\, P(x)$$

Entonces la prueba la podemos escribir de la siguiente forma

(1)	$\exists x\, (P(x) \wedge Q(x))$		Premisa
(2)	$\rightsquigarrow$		Inicio $\exists^{out}$
(3)		$P(a_0) \wedge Q(a_0)$	Supuesto
(4)		$P(a_0)$	$\wedge_1^{out}(3)$
(5)		$\exists x\, P(x)$	$\exists^{in}(4)$
(6)	$\exists x\, P(x)$		Fin $\exists^{out}$

Ejemplo 2.36. Probar que

$$\exists x\, P(x), \forall x\, (P(x) \to Q(a)) \vdash Q(a)$$

(1)	$\exists x\, P(x)$		Premisa
(2)	$\rightsquigarrow$		Inicio $\exists^{out}$
(3)		$P(a_0)$	Supuesto
(4)		$\forall x\, (P(x) \to Q(a))$	Premisa
(5)		$P(a_0) \to Q(a)$	$\forall^{out}(4)$
(6)		$Q(a)$	$\to^{out}(3,5)$
(7)	$Q(a)$		Fin $\exists^{out}$

Ejemplo 2.37. Probar que

$$\forall x\, (P(x) \to Q(x)), \forall x\, (Q(x) \to R(x)) \vdash \forall x\, (P(x) \to R(x))$$

(1)	$\forall x\, (P(x) \to Q(x))$		Premisa
(2)	$P(x) \to Q(x)$		$\forall^{out}(1)$
(3)	$\forall x\, (Q(x) \to R(x))$		Premisa
(4)	$Q(x) \to R(x)$		$\forall^{out}(3)$
(5)	$\rightsquigarrow$		Inicio regla fantasiosa
(6)		$P(x)$	Supuesto
(7)		$Q(x)$	$\to^{out}(6)$
(8)		$R(x)$	$\to^{out}(7)$
(9)	$P(x) \to R(x)$		Fin regla fantasiosa $\to^{in}(5,8)$
(10)	$\forall x\, (P(x) \to R(x))$		$\forall^{in}(9)$

2.4. Teoremas de Gödel de completitud e incompletitud. Al igual que en el caso de la lógica proposicional, cabe preguntarse si el sistema de deducción natural propuesto es equivalente a la deducción semántica, esto es, si es correcto y completo. Análogamente al caso proposicional tenemos las siguientes definiciones.

Definición 2.38. Sea $\mathcal{D}$ un sistema de deducción en Lógica de Primer Orden.

- Se dice que $\mathcal{D}$ es **correcto** si, para toda teoría $T \subseteq \mathbf{CFor}$ y toda $\psi \in \mathbf{CFor}$, $T \vdash \psi$ implica $T \models \psi$.
- Se dice que $\mathcal{D}$ es **completo** si, para toda teoría $T \subseteq \mathbf{CFor}$ y toda $\psi \in \mathbf{CFor}$, $T \models \psi$ implica $T \vdash \psi$.

Teorema 2.39 (Teorema de completitud)**.** *El sistema de deducción natural en lógica de primer orden es* ***correcto*** *y* ***completo****.*

Gracias al resultado anterior, podemos estar seguros de que todas las verdades semanticas pueden obtenerse mediante pruebas sintáctica y viceversa. No obstante, en lógica de primer orden tiene sentido preguntarse una noción un poco más general, conocida como Gödel completitud.

Definición 2.40. Una teoría T se dice **Gödel completa** si para toda fórmula cerrada $\psi \in \mathbf{CFor}$ se tiene $T \models \psi$ o $T \models \neg\psi$. Una teoría T se dice **inconsistente** si existe $\psi \in \mathbf{CFor}$ tal que $T \models \psi \wedge \neg\psi$. Una teoría que no es inconsistente se dice **consistente**.

Dicho de otro modo, una teoría T es Gödel completa si, dada cualquier afirmación $\psi \in \mathbf{CFor}$ bien formada, bien se tiene que ψ es cierta en T o bien $\neg\psi$ es cierta en T. Si esto ocurre quiere decir que T 'captura adecuadamente toda la información', con lo que no deja ninguna ambigüedad y todas las afirmaciones pueden ser juzgadas. Lamentablemente, Gödel demostró que, en general, esto no es así. De hecho, en cuanto una teoría es lo suficientemente compleja como para capturar las propiedades de suma y producto en los números naturales, está intrínsecamente enferma.

Definición 2.41. Un conjunto T se dice **recursivamente enumerable** si existe un algoritmo que enumere todos los elementos del conjunto T.

Observación 2.42. La definición de algoritmo es sutil, y hacerla precisa requiere trabajo. A nuestros efectos, diremos que un algoritmo es un programa implementable en un ordenador. De este modo, una teoría $T \subseteq \mathbf{CFor}$ es recursivamente enumerable si existe un programa que vaya imprimiento por pantalla todos las fórmulas de T, y ninguna más. La definición precisa de algoritmo utiliza una abstracción matemática de la noción de ordenador conocida como máquina de Turing.

Teorema 2.43 (Teorema de incompletitud de Gödel)**.** *Sea T una teoría consistente y recursivamente enumerable que capture la aritmética de los números naturales. Entonces T* ***no es Gödel completa****.*

Observación 2.44. Cuando hablamos de aritmética de los números naturales nos referimos a **su suma y su multiplicación**. Es decir, la teoría tiene que ser capaz de capturar las propiedades de suma y producto de los números naturales. Es posible construir teorías (por ejemplo, la aritmética de Presburger) que capturan únicamente la suma de números naturales y son consistenes y Gödel completas.

Observación 2.45. El teorema de incompletitud debe entenderse adecuadamente. En particular:

- **No** dice que haya verdades indemostrables. La lógica de primer orden es completa, por lo que todo lo que es verdad es demostrable sintácticamente. Lo que dice es que **hay afirmaciones que no son ni verdad ni mentira**.
- Obviamente, si $\mathcal{N} = (\mathbb{N}, \iota)$ es nuestro modelo usual de los número naturales, dada una fórmula $\psi \in \mathbf{CFor}$ o se tiene $\mathcal{N} \models \psi$ o se tiene $\mathcal{N} \not\models \psi$ (es decir, $\mathcal{N} \models \neg\psi$). Ese no es el problema que evidencia el teorema de incompletitud. En $\mathbb{N}$, una afirmación ψ o es verdad o es mentira, no hay medias tintas. Lo que dice el teorema de incompletitud es mucho más sutil.

  Supongamos que T es una teoría recursivamente enumerable que modeliza los números naturales. Por ejemplo, hay una formalización muy común de $\mathbb{N}$ conocida como los axiomas de Peano. Entonces existe una fórmula $\psi_0 \in \mathbf{CFor}$ tal que $T \not\models \psi_0$ y $T \not\models \neg\psi_0$. Entendamos esta afirmación un poco mejor. En mi modelo concreto de los números naturales, $\mathcal{N}$, sí ocurrirá que $\mathcal{N} \models \psi_0$ o $\mathcal{N} \models \neg\psi_0$. Supongamos que $\mathcal{N} \models \psi_0$ por simplicidad (en el caso contrario, el algumento es análogo pero al revés). ¿Cómo es posible que $T \not\models \psi_0$? Pues porque, aunque $\mathcal{N} \models \psi_0$, existe **otro modelo no estándar** de T, $\mathcal{M} \models T$, tal que $\mathcal{M} \not\models \psi_0$ (dicho de otro modo, $\mathcal{M} \models \neg\psi_0$). El modelo $\mathcal{M}$ cumple todas las propuedades de los números naturales (i.e. $\mathcal{M} \models \psi$ para todo $\psi \in T$) pero **su dominio no es** $\mathbb{N}$ sino algo distinto. A este $\mathcal{M}$ se le suele conocer como una **aritmética no estándar**. El estudio de los modelos de una teoría es una rama muy activa de la lógica conocida como **teoría de modelos**.
- La condición de que la teoría sea recursivamente enumerable es clave. Por ejemplo, si $\mathcal{N} = (\mathbb{N}, \iota)$ es el modelo **estándar** de $\mathbb{N}$, entonces podemos considerar la llamada **teoría de** $\mathcal{N}$

  $$\mathrm{Te}(\mathcal{N}) = \{\psi \in \mathbf{CFor} \,|\, \mathcal{N} \models \psi\}.$$

  Por construcción, esta teoría **si es Gödel completa** (si $\mathcal{N} \models \varphi$, entonces por definición $\varphi \in \mathrm{Te}(\mathcal{N})$ y, por tanto, $\mathrm{Te}(\mathcal{N}) \models \varphi$). La clave es que esta teoría **no es recursivamente enumerable**, es decir, **no existe un algoritmo que liste todas y cada una de las fórmulas que son ciertas en** $\mathbb{N}$.

2.5. Demostración automática. En esta sección, comentaremos cómo podemos utilizar las nociones de deducción sintáctica para diseñar un algoritmo capaz de realizar deducciones en lógica de primer orden, conocido como el **algoritmo de Robinson**.

Definición 2.46. Sea $\psi \in \mathbf{CFor}$ una fórmula cerrada. La **forma clausular** de ψ, denotada $\mathrm{FC}(\psi)$, es una teoría obtenida de la siguiente forma. Sea

$$\mathrm{Skolem}(\psi) = \forall x_1 \ldots \forall x_n \bigwedge_i C_i(x_1, \ldots, x_n)$$

la forma normal de Skolem de ψ, donde las C_i son clausulas (i.e. disjunción de átomos o negación de átomos). Entonces

$$\mathrm{FC}(\psi) = \{C_i(x_1, \ldots, x_n)\}_i.$$

Más aún, si $T\mathbf{CFor}$ es una teoría, su **forma clausular** es

$$\mathrm{FC}(T) = \bigcup_{\psi \in T} \mathrm{FC}(\psi).$$

Es decir, es la teoría formada por todas las cláusulas de las formas de Skolem de las fórmulas de T.

La estrategia de demostración automática en lógica de primer orden se basa en la misma herramienta que la lógica proposicional: la regla de resolución.

Proposición 2.47 (Regla de resolución)**.** *Para cualesquiera fórmulas $\psi, \varphi \in$ **For** sin variables comunes y todo átomo a, se tiene*

$$\psi \vee a, \varphi \vee \neg a \vdash \psi \vee \varphi.$$

Mientras que, en lógica proposicional, el uso directo de esta regla nos llevó al algoritmo de David-Putnam, en lógica de primer orden tenemos un problema evidente: el renombramiento. Para poder aplicar la regla de resolución con máxima generalidad, necesitamos tener una manera de **unificar** los nombres de las variables.

Definición 2.48. Una **substitución**, α, es una función

$$\alpha : \mathbf{Var} \to \mathbf{Term}$$

que asigna a cada variable $x \in \mathbf{Var}$, un término $\alpha(x) \in \mathbf{Term}$.

Operando recursivamente en los átomos, dada una fórmula $\psi \in \mathbf{For}$ y una substitución α, la **substitución** de α en ψ, denotada $\psi[\alpha]$, es el resultado de substituir todas las variables de ψ acorde a la substitución α. Dadas dos fórmulas, $\psi, \varphi \in \mathbf{For}$, diremos que son **unificables** si existe una substitución α tal que $\psi[\alpha] = \varphi[\alpha]$. En ese caso, α se llama el **unificador**.

Observación 2.49. Evidentemente, si ψ y φ son unificables, pueden serlo mediante multitud de unificadores. No obstante, siempre existe un unificador, esencialmente único, que es el 'más sencillo', el sentido de que cualquier otro puede obtenerse como composición de este unificador con otra substitución. Este unificador especial recibe el nombre de **Unificador Máximamente General** (UMG), y su cómputo eficiente es un área de investigación activa.

Usando unificadores podemos formular una regla de resolución más general, que nos permitirá deducir el análogo del algoritmo de Davis-Putnam en lógica de primer orden.

Proposición 2.50 (Regla de resolución con unificación)**.** *Sean $\psi, \varphi \in$ **For** dos fórmulas y a, b dos átomos sin variables en común. Si a y b son unificables con unificador α, entonces*

$$\psi \vee a, \varphi \vee \neg b \vdash (\psi \vee \varphi)\,[\alpha].$$

Observación 2.51. Si $\psi \vee a$ y $\varphi \vee \neg b$ tienen variables en común, también se puede resolver. Para ello, simplemente hay que cambiar los nombres de variable **antes** de resolver. Es decir, sean β_1 y β_2 dos substituciones tales que $(\psi \vee a)[\beta_1]$ y $(\varphi \vee \neg b)[\beta_2]$ no tienen variables en común. En este momento, podemos aplicar la regla de resolución obteniendo

$$\psi \vee a, \varphi \vee \neg b \vdash (\psi[\beta_1] \vee \varphi[\beta_2])\,[\alpha],$$

donde α es un unificador de $a[\beta_1]$ y $b[\beta_2]$.

Data: Teoría T de primer orden
Result: Si T es satisfacible o no
$X := \mathrm{FC}(T)$;
while ***not*** $\emptyset \in X$ **do**
  $\mathcal{R}(X) :=$ Conjunto de todos los resultados de aplicar resolución con unificación a todas las fórmulas de X que lo admiten;
  if $\mathcal{R}(X) = X$ **then**
    return T es satisfacible;
  end
  $X := \mathcal{R}(X)$;
end
return T es insatisfacible;

Algorithm 4: Algoritmo de Robinson

Proposición 2.52. *El algoritmo de Robinson es* ***semi-decidible para insatisfacibilidad****. Es decir, si T es insatisfacible (i.e. no existe una interpretación $\mathcal{A}$ tal que $\mathcal{A} \models T$) entonces el algoritmo de Robinson para y devuelve que T es insatifacible.*

Observación 2.53. Al contrario que el algoritmo de Davis-Putnam, el algoritmo de Robinson no es completamente decidible. Si T es insatisfacible, la Proposición 2.52 muestra que el algoritmo de Robinson para y arroja el resultado correcto. Por contra, si T es satisfacible, el algoritmo podría no parar nunca porque X crece indiscriminadamente, con lo que nunca devuelve el resultado correcto. De este modo, por mucho tiempo que haya pasado, no podemos estar seguros de que el algoritmo de Robinson no ha parado bien porque nunca parará porque T es satisfacible o porque aún no ha terminado de encontrar la contradicción.

Un ejemplo de teoría en la que el algoritmo no para es

$$T = \{P(a), \neg P(x) \vee P(f(x))\}.$$

Como se observa, en cada paso el algoritmo solo genera la familia infinita de fórmulas $P(f \circ f \circ \ldots \circ f(a))$ y no para nunca.

Observación 2.54. Al igual que en el caso de la lógica proposicional, la comprobación de insatisfacibilidad es suficiente para probar consecuencias lógicas. En efecto, dada una teoría T y $\psi \in \mathbf{CFor}$ una fórmula, se tiene

$$T \models \psi \Leftrightarrow T \cup \{\neg\psi\} \text{ es insatisfacible.}$$

3. Lógica difusa

3.1. Algunas paradojas de la lógica clásica. Consideremos el siguiente problema de lógica proposicional.

Si una persona no tiene ningún euro, entonces es pobre. Si una persona es pobre, y gana un euro, entonces sigue siendo pobre. ¿Es pobre una persona con un millón de euros?

En lógica clásica proposicional, escribiríamos este problema de la siguiente forma. Consideramos la familia de literales p_n que se interpreta como 'una persona que tiene n euros es pobre'. Entonces, la teoría que codifica la afirmación anterior es

$$T = \{p_0\} \cup \{p_n \to p_{n+1} \mid n \in \mathbb{N}\}.$$

Utilizando esta teoría, aplicando sucesivamente la regla del modus ponnens, en lógica clásica obtendríamos que

$$T \models p_{10^6}.$$

Luego, utilizando lógica clásica, una persona con un millón de euros es pobre.

Obviamente, el problema es que la afirmación 'Si una persona es pobre, y gana un euro, entonces sigue siendo pobre' no es categóricamente cierta. Según se van ganando más euros, esta afirmación pierde fuerza. El objetivo de la lógica difusa es resolver este problema.

3.2. Lógica difusa de Gödel-Dummett. En este apartado, estudiaremos una tipo de lógica muy extendido en Inteligencia Artificial, conocido como lógica de Gödel-Dummett. Las fórmulas con las que trataremos son **las mismas que en lógica proposicional**. El cambio clave reside en la asignación de verdad, que ahora puede tomar cualquier valor entre 0 y 1.

Definición 3.1. Una **evaluación de verdad de Gödel-Dummett** es una función

$$v_{GD} : \mathbf{For} \to [0,1]$$

tal que

$$\begin{aligned} v_{GD}(\psi \wedge \varphi) &= \text{mín}(v_{GD}(\psi), v_{GD}(\varphi)), \\ v_{GD}(\psi \vee \varphi) &= \text{máx}(v_{GD}(\psi), v_{GD}(\varphi)), \\ v_{GD}(\psi \to \varphi) &= \begin{cases} 1 & \text{si } v_{GD}(\psi) \leq v(\varphi) \\ v_{GD}(\varphi) & \text{si } v_{GD}(\psi) > v_{GD}(\varphi) \end{cases}, \\ v_{GD}(\neg\psi) &= \begin{cases} 1 & \text{si } v_{GD}(\psi) = 0 \\ 0 & \text{si } v_{GD}(\psi) > 0 \end{cases}. \end{aligned}$$

Al igual que en la lógica proposicional clásica, el valor de verdad de los literales determina completamente el valor de verdad de cualquier fórmula. Por ello, basta fijar $v(p)$ para todos los literales p.

Observación 3.2. Las asignaciones de verdad en lógica de Gödel-Dummett no toman valores binarios 0 o 1, sino cualquier valor intermedio. El valor $v_{GD}(\psi) \in [0,1]$ debe ser entendido como la cantidad de 'verosimilitud' que le damos a ψ.

Definición 3.3. Sea $T \subseteq \mathbf{For}$ una teoría y $\varphi \in \mathbf{For}$ una fórmula. Diremos que φ es **consecuencia semántica en la lógica de Gödel-Dummett**, y lo denotaremos $T \models_{GD} \varphi$, si para cualquier asignación de verdad de Gödel-Dummett, v_{GD}, tal que $v_{GD}(\psi) = 1$ para todo $\psi \in T$, se tiene que $v_{GD}(\varphi) = 1$.

Observación 3.4. Algunas propuedades típicas de la lógica clásica fallan en la lógica de Gödel-Dummett:

- $\not\models_{GD} \varphi \vee \neg\varphi$ (no se cumple el principio del tercero excluso). Por ejemplo, si $v_{GD}(\varphi) = 0{,}5$, se tiene que $v_{GD}(\varphi \vee \neg\varphi) = \text{máx}(0{,}5, 0) = 0{,}5$.
- $\not\models_{GD} \neg\neg\varphi \to \varphi$. Por ejemplo, si $v_{GD}(\varphi) = 0{,}5$, se tiene que $v_{GD}(\neg\neg\varphi) = 1$ (porque $v_{GD}(\neg\varphi) = 0$) y, por tanto, $v_{GD}(\neg\neg\varphi \to \varphi) = v_{GD}(\varphi) = 0{,}5$.
- $\not\models_{GD} \neg(\neg\psi \vee \neg\varphi) \to (\psi \vee \varphi)$.

Respecto al problema del millón de euros, supongamos que tomamos la evaluación de verdad de Gödel-Dummett dada por

$$v_{GD}(p_n) = 1 - \frac{n}{10^6}.$$

En ese caso, tendríamos que $v_{GD}(p_0) = 1$ y $v_{GD}(p_{10^6}) = 0$, lo cual tiene perfecto sentido. Sin embargo, tenemos que

$$v_{GD}(p_n \to p_{n+1}) = v_{GD}(p_{n+1}) = 1 - \frac{n+1}{10^6} \overset{n\to 10^6}{\to} 0.$$

Por este motivo, la lógica de Gödel-Dummett no es completamente satisfactoria, puesto que no captura bien la noción de implicación. El motivo es que la asignación de verdad para la implicación **no es continua**.

3.3. Lógica de Łukasiewicz. En este apartado, estudiaremos otra lógica difusa posible sobre las fórmulas proposicionales. El cambio proviene de que, ahora, cambiamos la definición de cómo la asignación de verdad actúa en la implicación. Como veremos, eso da lugar a una lógica más compleja, pero más natural en algún sentido.

Definición 3.5. Una **evaluación de verdad de Łukasiewicz** es una función

$$v_{Ł} : \mathbf{For} \to [0,1]$$

tal que

$$\begin{aligned} v_{Ł}(\psi \wedge \varphi) &= \text{mín}(v_{Ł}(\psi), v_{Ł}(\varphi)), \\ v_{Ł}(\psi \vee \varphi) &= \text{máx}(v_{Ł}(\psi), v_{Ł}(\varphi)), \\ v_{Ł}(\psi \to \varphi) &= \begin{cases} 1 & \text{si } v_{Ł}(\psi) \leq v(\varphi) \\ 1 - (v_{Ł}(\psi) - v_{Ł}(\varphi)) & \text{si } v_{Ł}(\psi) > v_{Ł}(\varphi) \end{cases}, \\ v_{Ł}(\neg\psi) &= 1 - v_{Ł}(\psi). \end{aligned}$$

Al igual que anteriormente, el valor de verdad de los literales determina completamente el valor de verdad de cualquier fórmula. Por ello, basta fijar $v(p)$ para todos los literales p.

Definición 3.6. Sea $T \subseteq \mathbf{For}$ una teoría y $\varphi \in \mathbf{For}$ una fórmula. Diremos que φ es **consecuencia semántica en la lógica de Łukasiewicz**, y lo denotaremos $T \models_{Ł} \varphi$, si para cualquier asignación de verdad de Łukasiewicz, $v_{Ł}$, tal que $v_{Ł}(\psi) = 1$ para todo $\psi \in T$, se tiene que $v_{Ł}(\varphi) = 1$.

Observación 3.7. En lógica de Łukasiewicz sigue fallando el principio del tercero excluso $\not\models_{Ł} \varphi \vee \neg\varphi$. Sin embargo, **sí se cumplen** otras propuedades esperadas, como la identidad de la doble negación $\not\models_{Ł} \neg\neg\varphi \to \varphi$ o las leyes de de Morgan, $\not\models_{Ł} \neg(\neg\psi \vee \neg\varphi) \to (\psi \vee \varphi)$.

Respecto al problema del millón de euros, hacemos lo mismo que antes y consideramos la evaluación de verdad de Łukasiewicz dada por

$$v_{Ł}(p_n) = 1 - \frac{n}{10^6}.$$

En ese caso, tendríamos que $v_{Ł}(p_0) = 1$ y $v_{Ł}(p_{10^6}) = 0$, lo cual tiene perfecto sentido. Además, tenemos que

$$v_{Ł}(p_n \to p_{n+1}) = 1 + v_{Ł}(p_{n+1}) - v_{Ł}(p_n) = 1 + \left(1 - \frac{n+1}{10^6}\right) - \left(1 - \frac{n}{10^6}\right) = 1 - \frac{1}{10^6},$$

lo cual tiene mucho más sentido, porque dice que la afirmación es cierta **independientemente del dinero que tengas**.

3.4. T-normas. Evaluación de verdad difusa.

Definición 3.8. Una **T-norma**, es una función

$$T : [0,1] \times [0,1] \to [0,1]$$

tal que:

- Conmutatividad: $T(a,b) = T(b,a)$ para cualesquiera $a, b \in [0,1]$..
- Monotonía: $T(a,b) \leq T(c,d)$ si $a \leq c$ y $b \leq d$.
- Asociatividad: $T(a, T(b,c)) = T(T(a,b), c)$ para cualesquiera $a, b, c \in [0,1]$.
- Identidad: $T(a,1) = a$

Definición 3.9. Dada una T-norma, T, y $a, b \in [0,1]$, definimos su residuo como

$$R(a,b) = \sup \{c \,|\, T(c,a) \leq b\}\,.$$

Observación 3.10. Si la T-norma es continua, lo anterior quiere decir que $R(a,b) = c$ si y solo si $T(c,x) = b$.

Definición 3.11. Una fórmula de T-lógica proposicional es una fórmula construida al igual que en la lógica proposicional, pero utilizando como operadores lógicos los símbolos:

- &: Conjunción difusa fuerte.
- $\to$: Implicación difusa.

Asímismo, añadimos un literal denotado por $\bot$, denominado **contradicción**.

Definición 3.12. Sea T una T-norma. La **evaluación de verdad asociada a** T es una función

$$v_T : \mathbf{For}_T \to [0,1]$$

tal que

$$\begin{aligned} v_T(\psi \& \varphi) &= T(v_T(\psi), v_T(\varphi)), \\ v_T(\psi \to \varphi) &= R(v_T(\psi), v_T(\varphi)), \\ v_T(\bot) &= 0. \end{aligned}$$

Utilizando estas reglas, podemos recuperar el significado de las fórmulas clásicas escribiendo los operadores lógicos usuales en términos de los operadores T-lógicos como:

- $v_T(\psi \wedge \varphi) = v_T\left(\psi \& (\psi \to \varphi)\right)$.
- $v_T(\psi \vee \varphi) = v_T\left(\neg(\neg\psi \wedge \neg\varphi)\right)$.
- $v_T(\neg\psi) = v_T\left(\psi \to \bot\right) = R(v_T(\psi), 0) = \sup\{c \,|\, T(v_T(\psi), c) \leq 0\}$.

Observación 3.13. Las lógicas de Gödel-Dummet y Łukasiewicz son, de hecho, casos particulares de lógicas difusas asociadas a T-normas. Estas se obtienen con las siguientes T-normas:

1. Gödel-Dummett: $T(a,b) = \text{mín}(a,b)$.
2. Łukasiewicz: $T(a,b) = \text{máx}(a + b - 1, 0)$.

Otra elección común es $T(a,b) = a \cdot b$, que da lugar a la llamada lógica producto, que está relacionada con la independencia de eventos probabilísticos.

3.5. Lógica Difusa de Primer Orden y sus aplicaciones a Inteligencia Artificial. Consideremos ahora la extensión a Lógica de Primer Orden de los conceptos de lógica difusa. Al igual que pasó con la lógica clásica, la diferencia fundamental radica en que, ahora, debemos lidiar con conjuntos (en nuestro caso, difusos).

Definición 3.14. Sea Ω un conjunto. Un **subconjunto difuso** de Ω es una función $\mu : \Omega \to [0,1]$.

Observación 3.15. Los subconjuntos clásicos $A \subseteq \Omega$ son conjuntos difusos de forma natural considerando su función indicatriz, μ_A, dada por $\mu_A(x) = 1$ si $x \in A$ y $\mu_A(x) = 0$ si $x \notin A$.

Definición 3.16. Sea L un lenguaje de lógica de primer orden. Un **modelo difuso** para L es una tupla $\mathcal{A} = (A, \iota)$, donde A es un conjunto, llamado el **dominio**, y ι es una asignación tal que:

- A toda constante c de L, asigna un elemento $c^{\mathcal{A}} \in A$.
- A toda función f de L de aridad n, asigna una función $f^{\mathcal{A}} : A^n \to A$.
- A toda relación R de L de aridad n, asigna un subconjunto difuso $R^{\mathcal{A}} : A^n \to [0,1]$.

Definición 3.17. Sea L un lenguaje y $\mathcal{A}$ un modelo difuso para L. Escogida una lógica difusa, entonces tenemos una función

$$v_{\mathcal{A}} : \mathbf{CFor} \to [0,1]$$

definida de la siguiente forma:

- Si $\psi = R(t_1, \ldots, t_n)$, con t_i términos parametrizados por $\mathcal{A}$, entonces hacemos
$$v_{\mathcal{A}}(\psi) = R^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}) \in [0,1],$$
visto como un conjunto difuso.
- Para el resto de símbolos lógicos, la interpretación se realiza recursivamente como determina la lógica difusa.

Definición 3.18. Sea $\varphi(x_1, \ldots, x_n) \in \mathbf{For}$ una fórmula de primer orden con variables libres. Dado un modelo $\mathcal{A}$, la **verosimilitud** es la función

$$\mathcal{L}_{\mathcal{A}} : A^n \to [0,1]$$

dada por $\mathcal{L}_{\mathcal{A}}(a_1, \ldots, a_n) = v_{\mathcal{A}}(\varphi(a_1, \ldots, a_n))$.

Ejemplo 3.19. Una aplicación omnipresente de la lógica difusa de primer orden son los llamados **controladores difusos**. Estos no son más que evaluaciones de la verosimilitud de una fórmula con variables libres en la lógica de Gödel-Dummett. Ilustremos esta idea con un ejemplo.

Consideremos el lenguaje con relaciones unarias siguientes:

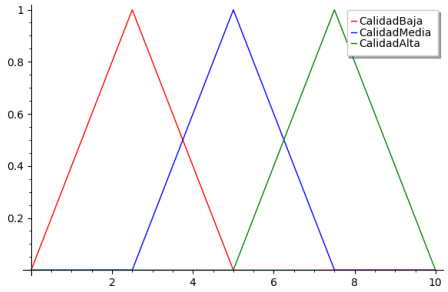
$$\text{CalidadBaja}, \text{CalidadMedia}, \text{CalidadAlta}, \text{ServicioMalo}, \text{ServicioMedio}, \text{ServicioBueno},$$

$$\text{PropinaBaja}, \text{PropinaMedia}, \text{PropinaAlta}.$$

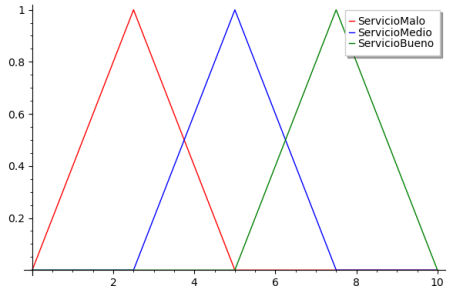
Queremos modelizar los criterios para determinar la propina tras acudir a un restaurante. Para ello, consideramos la fórmula

$$\begin{aligned}\varphi(x) = &((\text{CalidadAlta}(c) \vee \text{ServicioBueno}(s)) \wedge \text{PropinaAlta}(x)) \\ &\vee (\text{ServicioMedio}(s) \wedge \text{PropinaMedia}(x)) \\ &\vee ((\text{CalidadBaja}(c) \vee \text{ServicioMalo}(s)) \wedge \text{PropinaBaja}(x))\,.\end{aligned}$$

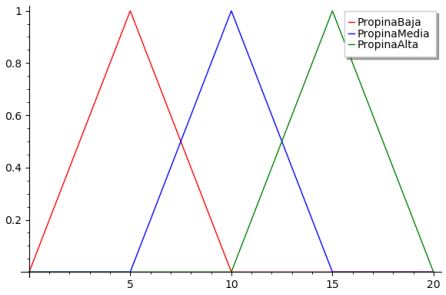
Asimismo, tomamos la interpretación $\mathcal{A}$ que toma como dominio para la calidad c y el servicio s el intervalo $[0, 10]$ (puntuación del 0 al 10). Además, el dominio para la propina es el intervalo $[0, 20]$ (propina entre el 0 % y el 20 % de la cuenta). Respecto a los conjuntos difusos correspondientes a las relaciones anteriores, la interpretación $\mathcal{A}$ asigna las funciones mostradas en la Figura 1.



(A) Conjuntos difusos de "Calidad"



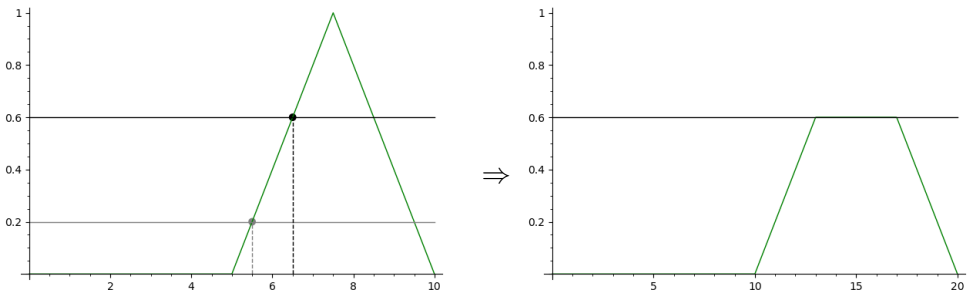
(B) Conjuntos difusos de "Servicio"



(C) Conjuntos difusos de "Propina"

FIGURA 1. Conjuntos difusos de las interpretaciones de las relaciones del lenguaje.

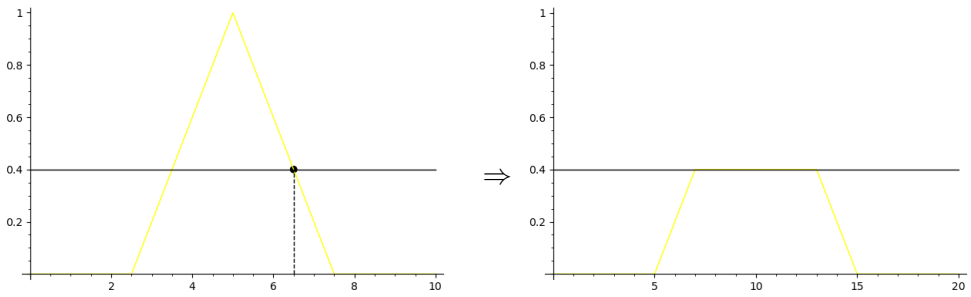
Ahora, consideremos que, en esta interpretación, tenemos que la calidad ha sido $c^{\mathcal{A}} = 5{,}5$ y el servicio $s^{\mathcal{A}} = 6{,}5$. Entonces podemos calcular la verosimilitud de cada una de las tres reglas que componen φ. La representación gráfica de la primera regla se muestra en la Figura 2, la de la segunda regla en la Figura 3, y la de la tercera en la Figura 4. Obsérvese que, en todos los casos, hay que calcular la función mínimo entre la parte constante que incluye c y s y la parte variable que depende de x.



(A) CalidadAlta(c) $\vee$ ServicioBueno(s)

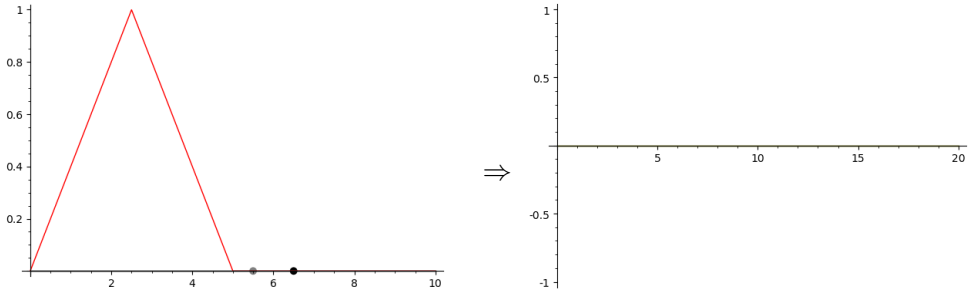
(B) Verosimilitud de la primera regla

FIGURA 2. Verosimilitud de la primera regla de la fórmula $\varphi(x)$, a saber, (CalidadAlta(c) $\vee$ ServicioBueno(s)) $\wedge$ PropinaAlta(x).



(A) ServicioMedio(s) (B) Verosimilitud de la segunda regla

FIGURA 3. Verosimilitud de la segunda regla de la fórmula $\varphi(x)$, a saber, ServicioMedio(s) $\wedge$ PropinaMedia(x).



(A) CalidadBaja(c) $\vee$ ServicioMalo(s) (B) Verosimilitud de la tercera regla

FIGURA 4. Verosimilitud de la tercera regla de la fórmula $\varphi(x)$, a saber, (CalidadBaja(c) $\vee$ ServicioMalo(s)) $\wedge$ PropinaBaja(x).

Con estos resultados podemos calcular la verosimilitud global de la fórmula φ, denotada $\mathcal{L}$. Para ello, para cada valor posible de la propina a, calculamos el máximo entre las funciones de las Figuras 2, 3 y 4. El resultado se muestra en la figura 5

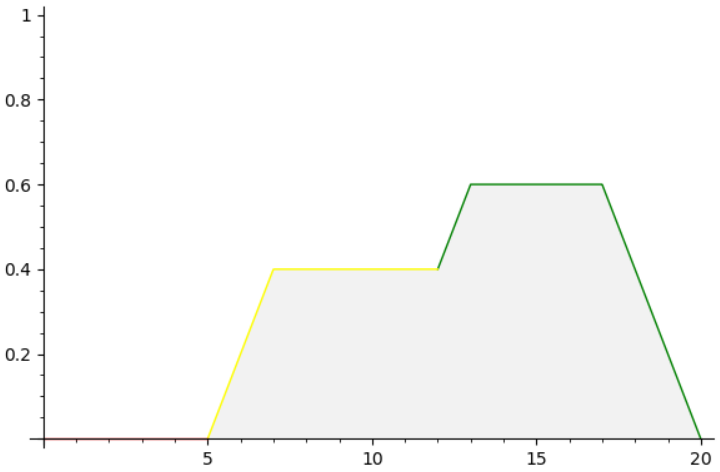


FIGURA 5. Función de verosimilitud de φ, $\mathcal{L}(a)$.

Este gráfico representa la verosimilitud que de que, finalmente, se aporte cada una de las propinas posibles. En este punto, si queremos tomar una decisión (i.e. cuánto dinero sacar de la cartera) necesitamos un criterio que resuma esta función de verosimilitud en un único valor.

Típicas elecciones son considerar el punto máximo de la verosimilitud (aunque, en este caso, nos produciría una ambigüedad puesto que hay todo un segmento de máximos) o el centro de gravedad de la zona sombreada en gris (i.e. la mediana de la distribución asociada). Este proceso se conoce como "defuzzificación", y es crítico para la eficacia del método de control difuso.

Referencias

Departamento de Álgebra, Geometría y Topología. Facultad de Ciencias Matemáticas. Universidad Complutense de Madrid

Email address: angelgonzalezprieto@ucm.es